DAMAGE IDENTIFICATION IN A REAL STRUCTURE USING RESONANT AND ANTI-RESONANT FREQUENCIES

THESIS

Douglas E. Gaeta, Captain, USAF

AFIT/GA/ENY/00M-02

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GA/ENY/00M-02

Damage Identification in a Real Structure Using Resonant and Anti-Resonant Frequencies

THESIS

Presented to the Faculty of the Graduate School of Engineering and Management

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Astronautical Engineering

Douglas E. Gaeta, B.S.

Captain, USAF

March, 2000

AFIT/GA/ENY/00M-02

Damage Identification in a Real Structure Using Resonant and Anti-Resonant Frequencies

Douglas E. Gaeta, B.S.

Captain, USAF

Approved:

[illegible]	09 MAR 2000
Lt Col Jeffrey Turcotte Ph.D.	Date
Thesis Advisor	
Bradley S. Liebst	09 MAR 2000
Doctor Bradley Liebst Ph.D.	Date
Committee Member	
[illegible]	09 Mar 00
Captain Gregory Agnes Ph.D.	Date
Committee Member	

Preface

I would like to acknowledge those who have helped me in the completion of this thesis. Thanks go to Lieutenant Colonel Jeffrey Turcotte for his expertise, guidance, and encouragement in helping make this work possible. Captain Al Arb's help in generating the computer code necessary for damage detection proved to be quite valuable. Additionally, I would like to thank Mr. Jay Anderson and his team of technicians that were always there to help me and the other students with lab set-up and equipment integration problems.

Thanks to my family back in Amesbury, MA and my fiancee Laylah. Although they were not here with me, their endless support and encouragement was invaluable to me throughout my entire AFIT experience. Finally, thanks to my father, who is longer with us. The dedication, hard-work, and discipline he instilled in me has been the foundation of all my successes.

Douglas E. Gaeta

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AFIT/GA/ENY/00M-02

Abstract

There exists a need to remotely monitor the structural integrity of large space structures without costly manned missions. This work focused on further damage detection characterization of the Air Force Institute of Technology's Flexible Truss Experiment (FTE). The FTE is intended to be representative of a large space structure.

Several damage detection algorithms were developed and tested for the FTE using 112 different damage conditions and the one undamaged condition. The algorithms were trained using two frequency response functions (FRFs) from each damage case and then tested using the same two FRFs, but from newly acquired data. A data reduction technique from the field of speech recognition was adapted for this damage detection application. The data was reduced by greater than an order of magnitude, via a discrete, point-by-point, integration process in both training and testing. As shifts in both the resonant and anti-resonant frequencies were caused by the damage, another damage detection algorithm was developed that extracted the resonant and anti-resonant frequencies from the two FRFs. This algorithm vectorized the frequencies of the peaks and valleys in the FRFs for comparison.

Over 99% accuracy was obtained using both the adapted speech recognition method and the resonant and anti-resonant frequencies method. However, only 44% accuracy was achieved by training the resonant and anti-resonant frequencies method with synthetic, Finite Element Model generated data, and testing it with measured data.

I. Introduction and Background

1.1 Introduction

Considerable research and effort over the last few decades has taken place in the field of damage detection and damage classification in structures. These structures include, but are not limited to, space structures and off-shore oil platforms. It is the former case that is of interest in this work.

Researchers have shown that reliable, fast, and efficient damage identification techniques based on the vibrational characteristics of structures provide good to moderate results. The motivation for this work stems from a need to accurately monitor the integrity of a spacecraft structure remotely, i.e., without requiring a manned mission and/or a spacewalk, to conduct a visual inspection. In many instances, a visual inspection is impossible. For example, a satellite at geostationary orbit cannot be reached via a Space Shuttle Mission.

This thesis explores several damage detection methodologies used on a real structure that are based on vibrational differences between undamaged and damaged states.

1.2 Problem Statement

Many damage detection methods rely upon mode shape comparison between an unknown, structure damage state and an undamaged, baseline state. In the case of the mode shapes, several measurements must be taken over the structure. In 1985, Yuen (34) proposed a method by which the mode shapes were used to compute the reduction in stiffness. In 1992, Oseguada (23) was unsuccessful in his attempt to correlate mode shape changes to damage in an offshore platform.

Damage detection results based on mode shapes have been moderate and, in the second example cited above, poor. More importantly, their inherent reliance upon many measurement locations has the adverse affect of increasing the complexity and weight of the spacecraft as a whole, with more output sensors required. Therefore, the number of sensors should be minimized even though this will decrease mode shape information. For these reasons, mode shapes will not be explored as a means for detecting damage in this work. As piezoelectric technologies continue to mature, a more viable mode shape technique may be used for space based structural damage detection in the future.

The motivation for this work stems from a need to improve damage detection methods. A method to detect varying degrees of damage is desired. Cobb (5) created an Assigned Partial Eigenstructure (APE) algorithm for damage detection on the Flexible Truss Experiment (FTE). Slightly damaged cases were tested, and approximately 40% accuracy was achieved. Cobb computed natural frequencies and mode shapes based upon eight FTE accelerometer measurements to detect damage. Swenson (29) was only able to detect significant FTE damage; however, he did so using only two accelerometer measurements. This work will attempt to detect varying FTE damage with minimal accelerometer data. This would be a significant improvement over the works of both Cobb (5) and Swenson (29).

1.3 Research Objectives

The objectives of this research are to improve current and develop new damage detection methodologies using the Air Force Institute of Technology's (AFIT) FTE. The FTE is a six meter tall truss made primarily of aluminum that also contains 32 identical, bolt-on, diagonal members made of Lexan that can be easily removed and replaced. A more detailed discussion of the FTE will be provided in Chapter II. A portion of the work will focus on improving and increasing the damage detection capabilities via the method outlined by Swenson (29), which utilizes a data reduction algorithm adapted from the field of speech recognition. The other work will be the creation of a new damage detection algorithm based upon the measured resonant and anti-resonant frequencies of the FTE. A method based solely upon resonant frequencies will be created and compared to a similar method based upon both resonant and anti-resonant frequencies.

All methods will seek the capability to detect partial damage of the FTE diagonals and multiple damaged diagonals, thus representing a more than three fold increase in the damage detection capability as compared to the method Swenson proposed (29). It is the objective of this work to demonstrate that the damage detection methods proposed are viable solutions for remote monitoring of spacecraft structural integrity. However, the reader should be aware that the major drawback of the methods proposed are its assumption of a known number of likely damage scenarios. In a real structure, an infinite number of damage cases are possible, with some more likely than others.

In another on-going effort, a Finite Element Model (FEM) tuning method is being created for the FTE by Jones (17). The purpose of this work is to simulate the FEM dynamic behavior as close as possible to the true measured dynamic behavior. Coupled with the proposed damage detection algorithms, the FEM can be used to simulate any possible damage case and thereby expand the damage envelope that otherwise would not be possible with the experimental structure without destructive testing. Assuming the FEM generated data is close enough to the real structure data, damage detection pattern classifiers can be based and trained on artificially created vibration profiles. This has the dual benefit of decreasing lab data acquisition time and increasing the number of possible damage cases. Additionally, a FEM that closely resembles the real structure's vibrational profile can help ameliorate the problem of assuming a finite number of damage cases. The FEM can be modified in a iterative fashion until it closely matches the measured vibration profile, assuming, of course, that the FEM itself is accurate. This would provide considerable damage anomaly insight to a team of structural engineers trying to diagnose structural integrity; however, this type of analysis would not be practical for day-to-day spacecraft operations in the field.

1.4 Thesis Scope

This thesis will be limited to development of damage detection algorithms based solely upon the different vibrational characteristics of the FTE. Measurements will be taken at two locations on the FTE with their axes being orthogonal for 112 possible damage cases and one undamaged case, also known as the baseline configuration.

The damage detection algorithms will be created in such a fashion as to allow any person to randomly select any of the 112 damage states to be tested in the experimental set-up, which includes the FTE and associated data acquisition equipment.

1.5 Thesis Outline

An outline of this thesis is as follows. The remainder of this chapter provides a brief description of literature addressing various damage detection methodologies and theories. The following is a list of the damage detection methods addressed:

- Artificial Neural Networks
- Mode Shape Shifts
- Natural or Resonant Frequency Shifts
- Resonant Frequency and Anti-Resonant Frequency Shifts

Chapter II contains the description of the experimental set-up and how data was gathered and stored from the FTE. Included in this chapter are physical illustrations of the hardware utilized, in addition to their purpose. Chapter III outlines the damage detection methodologies developed, including justification for use in this work. In Chapter IV, a description of how the FTE was tested and the results of those tests using all damage detection methodologies outlined is provided. Finally, Chapter V provides conclusions and recommendations.

1.6 *Structural Damage Detection Methods*

Although damage detection methods vary greatly, most, if not all, rely on the dynamic response of some structure or model in their characterization of the damage status of a structure. Figure 1.1 illustrates the overall damage detection processes developed in this work, (25). A discussion of the various methodologies found in the literature will be presented in this section.

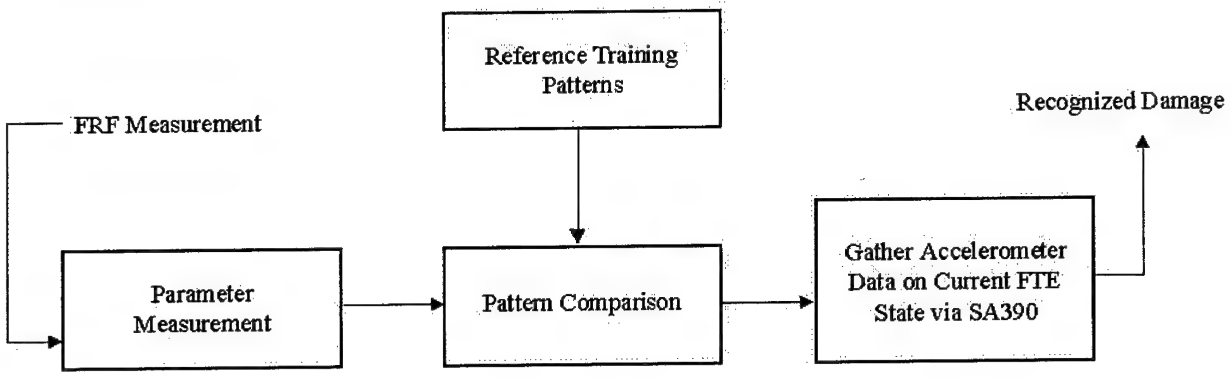

Reference Training Patterns
FRF Measurement
Recognized Damage
Parameter Measurement
Pattern Comparison
Gather Accelerometer Data on Current FTE State via SA390

Figure 1.1 Block Diagram For Damage Detection Pattern Recognition Process

1.6.1 Artificial Neural Networks.

Inspired by the neuronal architecture of the brain, an artificial neural network (ANN) is a computational model that is composed of simple processors, called neurons or nodes, and numerous connections between them (10). These connections are known as weights with the weight being determined by the strength of the bond between the two neurons. These neural networks are capable of self-organizing and learning. Learning or training of the ANN typically involves adjustment of these weights to produce the desired output. For a graphical illustration of ANNs, see Swenson (29).

Due to the layered structure of most ANN's, they are commonly referred to as a Multi-Layered Perceptron (MLP), (33). Training of the MLP's is usually conducted in what is termed loosely as supervised learning (33). In supervised learning, the MLP is 'told' the expected output in addition to the input data. Conversely, unsupervised learning occurs when the MLP is not told what the expected output is, and the MLP must determine similarities among the input patterns and determine the weights on its own. This supervised and unsupervised learning is an iterative process, and it is continued until the desired outcome is produced via back-propagation of the errors through the network via neuron interconnection weight adjustment.

Atalla used ANNs to perform FEM updating and subsequent damage detection on both a cantilever beam and a flexible frame structure (4). Atalla's finite element model estimated the modal parameters being updated quickly and accurately without the need of measuring all degrees of freedom. The technique implemented avoided the use of mode shape expansion, and it was designed to be easily incorporated in an adaptive control scheme (4).

In 1991, Kudva, et al. (19) used a back-propagation neural network to identify damage in a plate stiffened with a 4 X 4 array of bays. Damage was simulated by drilling holes of various diameters in the plate at the bay's center. The MLP successfully determined the damaged bay, but it had difficulty in predicting the diameter of the hole. The correct hole size was diagnosed with only 50% accuracy.

In 1993, Worden, et al. (32) used a backpropagation neural network to identify damage in a twenty-member structure. Complete member removal was used to model damage. The ANN was trained with strain data from eight of the members that were to be tested for damage identification. The researchers designed a three-hidden-layer ANN design with 12, 12, and 8 hidden nodes, respectively. The network was trained with FEM generated data, and it was then tested with data from the real structure. Exact accuracy numbers were not given, but the researchers claimed that the ANN usually identified the damage member.

In 1993, Elkordy, et al. (10) used a MLP for damage detection of a five-story steel frame mounted on a shaking-table platform. The network was trained with changes in the vibration spectra, i.e., resonant frequencies, from the undamaged state to the damaged state instead of just the absolute values of the spectra. Damage was simulated by reducing stiffness from 30 to 70% in the bottom two stories, and the MLP was reasonably successful in identification of the damage even when trained with partially inaccurate data samples.

Not surprisingly, the amount of training required increased as the data samples became more inaccurate.

In 1994, Stephens and VanLuchene (27) used a backpropagation network to identify damage in a one-tenth scale model of a reinforced concrete structure. The researchers introduced cracks into the structure. The researchers then used displacement data, dissipated energy in the structure, and stiffness degradation to train the network. The neural network correctly identified damage 78% of the time. In addition, the network was successively implemented for damage detection on a government services building in an area prone to earthquakes.

Leath and Zimmerman (20) used a MLP to identify damage in a four-element, cantilevered beam FEM. The training algorithm developed by the researchers was designed to iteratively change the architecture of the network. To this end, a network was created that had a minimal number of hidden nodes. This had the effect of exactly fitting the network to the data, which would be undesirable if there is noise present. Noisy data tended to propagate through the network causing errors. Damage in the model was simulated by reducing Young's modulus by up to 95%. The network used the first two bending frequencies to determine the level of damage in the beam. The training algorithm used by the researchers limited the network to consider only the first two bending frequencies. Only 35% damage detection accuracy was achieved.

Manning (21) proposed a method based upon active member transfer function characteristics and artificial neural networks on a 25-bar transmission tower with active members. Damage was simulated by reducing the cross-sectional area of a member. Manning trained the network with active member transfer function poles and zeros. The MLP consisted of 40 inputs, two hidden layers with seven and five nodes, and four outputs. Manning's network was able to identify the general area of the damage in the tower, but it was not able to identify the degree of damage within 10%.

The examples cited above utilizing MLP's and others over the last 10 years provided the basis by which Swenson (29) tested the Flexible Truss Experiment. Swenson developed and tested a MLP in the LNKnet simulation program. Several MLP variations were tested, and the configuration that gave the best damage detection results was used. This MLP consisted of 20 input nodes, 25 hidden nodes, and 33 output nodes with only one hidden layer. Using data to train the MLP from the FEM generated data that was closest in matching the FTE's dynamic behavior, the network was able to identify the damage 54% of the time.

Due to Swenson's limited success with the MLP's on the FTE, and the fact that better results were achieved with other methods, MLP's will not be used in this work. In addition, the selection of the number of nodes and hidden layers appears to be quite arbitrary in the literature cited with no general guidelines in their selection. These design factors seem to be system specific and directly correlate with the amount of training and damage detection accuracy.

1.6.2 Modal Analysis Methods.

Modal analysis methods have been employed in damage detection techniques for over thirty years. Damping factors, resonant frequencies, and mode shapes can be found via modal analysis methods. One particular method, Eigensystem Realization Algorithm (ERA), will be discussed in Chapter 3.

1.6.2.1 Mode Shape Shifts.

Typically, natural frequencies or resonant frequencies can be measured with a resolution of 0.1% in a lightly damped structure, whereas, typical mode shape errors are at least 10%, (11). Additionally, mode shapes are usually insensitive to damage. As stated previously, mode shapes rely upon many sensor locations for measurement. For these reasons mode shape methods will be excluded from the discussion that follows on damage detection methods.

1.6.2.2 Resonant Frequency Shifts.

One of the early damage detection works occurred in 1977 when Vandiver (30) examined the changes in the resonant frequencies associated with first two bending modes and first torsional mode of an offshore light station tower. Vandiver simulated the dynamic behavior of the tower and showed that the sloshing of fluid in tanks, which changes the tower's effective mass, accounted for only 1% change in the natural frequencies of the modes considered. Vandiver also showed that removing structural members produced higher resonant frequency changes, thereby validating damage detection via changes in the resonant frequencies.

In 1998, Agneni, et al. (2) utilized natural frequencies and Frequency Response Functions (FRF) in their FEM updating procedure of a composite cantilevered beam. This structure was also used for damage identification. Damage was simulated and measured at various locations in the beam to determine stiffness changes in the FEM updating process. Once the researchers felt the FEM model closely matched the behavior of the real structure, a damage identification technique was created that utilized changes in both the resonant frequencies and the FRF's. Moderately good results were obtained, again proving resonant frequency shifts can be used in damage detection.

James, et al. (16) developed a damage detection procedure for NASA's Vertical Stabilizer Assembly (VSA) used on the space shuttle by calculating the first Ritz Vectors that were derived from the FRFs. For more information on Ritz Vectors, consult Meirovitch (22). Nominal, undamaged, operating, FRF bounds were computed for the VSA. Damage was characterized by deviations from these bounds. The Ritz Vector, derived from FRF data, served to reduce the amount of data, while at the same time retaining the unique state of the VSA, i.e., damaged or undamaged. Eight different damage cases were modeled. Exact accuracy numbers were not provided, but the researchers claim their FRF based approach is sensitive to damage. Another aspect of this research showed that the accelerometer on the extreme tip of one of the structure's fins showed the most variability in its FRFs. It also makes intuitive sense that the most extreme points will display more dynamic variability when the structure is damaged.

Williams et al. (31) used a FRF based approach for damage detection on a steel frame. Natural frequencies were extracted within 0.15% accuracy for the structure. The researchers found that using percentage changes in the natural frequencies yielded better damage detection results than those obtained using absolute changes in natural frequencies. Higher frequency modes would dominate the case where absolute changes were used. When percentage changes in frequency were used, the lower frequency modes made more of a contribution to the overall damage profile. The researchers were able to achieve at least 95% damage detection accuracy.

A Frequency Response Function Assignment Method was used by Schulz, et al. (26) for damage detection on a model of an 18 member, simply-supported, truss. The researchers began with an analytical model of the truss that was tuned using the measured natural frequencies. Tuning is the process by which the analytical model is adjusted until its dynamic response, which in this case is manifested by the FRF, matches the measured dynamic response. After tuning, damage was identified via changes in the damping and stiffness matrices. On the real structure, this technique would require the use of a large number of sensors, or a large movable sensor system. In testing of the model, the researcher's were able to successfully diagnose damage (100%) if data from all nodes in the immediate vicinity of the damage was used. Otherwise, the damage was misclassified as being spread over the entire structure.

1.6.2.3 Resonant and Anti-Resonant Frequency Shifts. Afolabi (1) proposed an anti-resonance technique for detecting damage in a modeled cantilever beam. An anti-resonant frequency occurs when the magnitude of the frequency response approaches zero for a degree of freedom. An anti-resonant frequency can be seen at approximately 13.8 Hz in the FTE's undamaged configuration, see Figure 1.2.

The idea behind Afolabi's research is that anti-resonant frequencies vary with measurement location, whereas, resonant frequencies do not vary with measurement location in any one particular damage state. Although a damage detection technique was not created, Afolabi simulated local loss of stiffness and mass. In both cases, it was shown that the resonant frequencies did not change at various measurement points along the beam. The anti-resonant frequencies did change, as expected, at points away from the damage, but they did not change at the point where damage occurred. Afolabi (1) showed that changes in mass and/or stiffness at a certain location do not change the receptance, or acceleration at that point; therefore, the anti-resonant frequency is insensitive to changes in mass and stiffness.

The research above shows that both FRF, and resonant & anti-resonant frequency changes are a viable means for damage detection in real structures. FRFs will be gathered from several configurations of the FTE. This measured FRF data will form the basis for

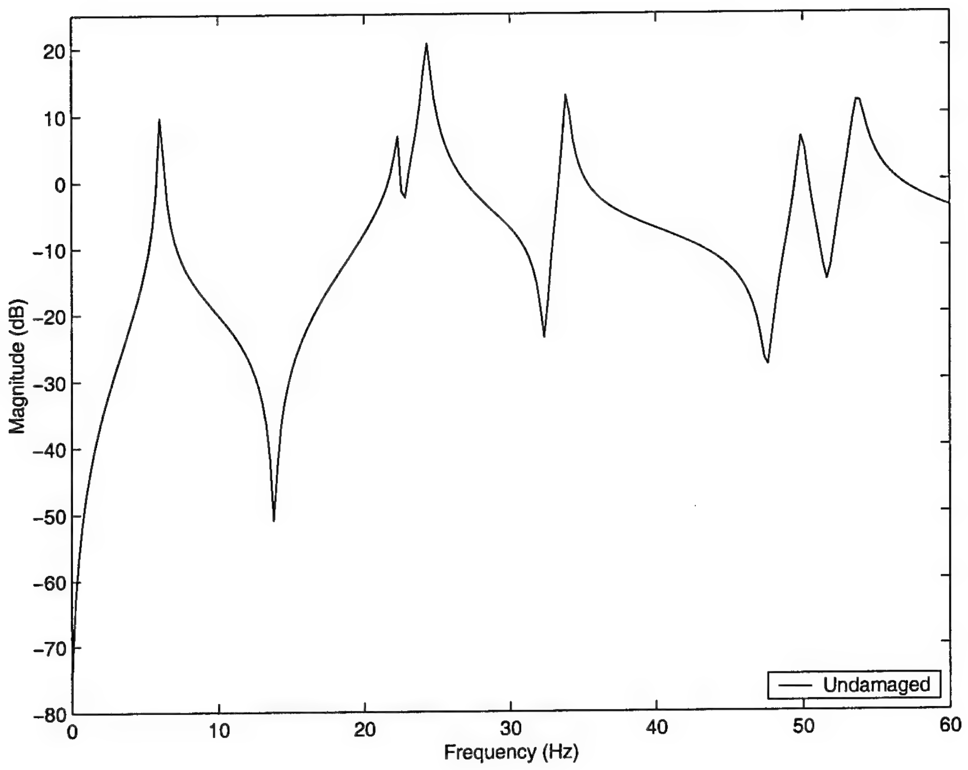


Figure 1.2 FTE Undamaged, X-Direction FRF

pattern classifiers that will be developed and tested for accuracy in damage detection. If the pattern classifiers prove successful from data trained by the real measured data, the pattern classifiers could be trained with synthetic FTE FRFs generated by a FEM and tested on the real structure. Jones (17) is developing a high-accuracy FEM of the FTE. The whole point in developing an accurate FEM is to eliminate the burden of gathering real FRF data from the FTE and to thereby train the damage detection pattern classifiers with synthetic data.

II. Experimental Setup & Data Acquisition

2.1 Introduction

This chapter focuses on the test specimen, the Flexible Truss Experiment, and its associated test equipment. The experimental setup will be discussed in detail along with how the vibration data (FRFs) were generated, gathered, and saved.

2.2 Experiment Hardware

The following sections discuss the FTE and the equipment used to gather the FTE's dynamic response.

2.2.1 Flexible Truss Experiment. The six-meter FTE came from the Wright Laboratories Large Space Structures Technology Program, where it was used as a testbed for active vibration control (13). The original FTE used by Wright Laboratories was 12 meters long and consisted of four, 3-meter long sections with four bays per section. Due to facility constrains, two sections were removed when the structure was brought to the Air Force Institute of Technology (AFIT). The FTE is a truss comprised of welded, hollow, aluminum alloy members. In addition to the aluminum members, there are 32 bolt-on, Lexan diagonal members-four diagonal members per bay. Figure 2.1 displays a typical FTE bay.

The FTE is bolted to a 1" thick aluminum plate at its base. The aluminum plate, in turn, is bolted to the concrete floor with five large bolts. These bolts are secured with fasteners commonly referred to as spreaders. Figure 2.2 shows the aluminum plate and fasteners.

2.2.2 Electromagnetic Shakers. Two electromagnetic shakers, shown in Figure 2.3, were used simultaneously at the top of the FTE to excite it in two mutually

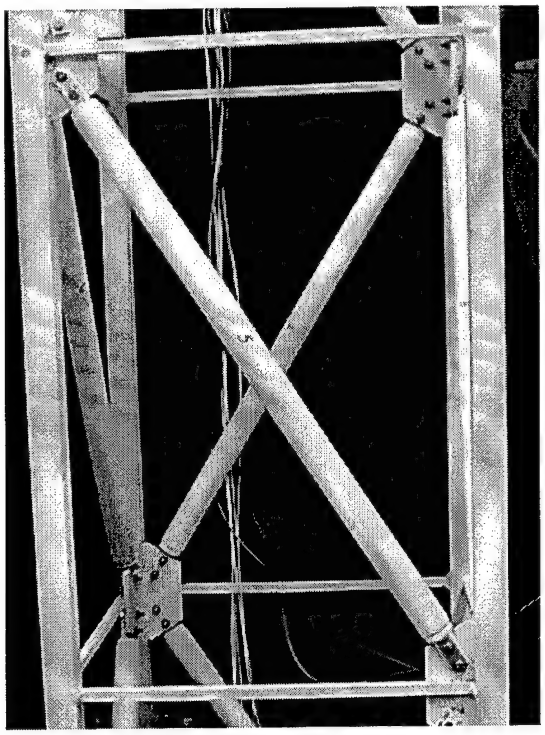


Figure 2.1 Typical FTE Bay

orthogonal directions. Swenson (29) found that leaving both actuators on, as opposed to operating them separately, resulted in no shift in modal frequencies greater than 0.25 Hertz, nor were there any changes in amplitudes. Leaving both actuators on halved the data acquisition time during the experiment. The shakers were bolted directly to the top of the FTE. Although an external excitation mechanism is desirable in dynamic testing and for finite element modeling analysis purposes, it was felt that mounting the shakers directly to the structure would be typical in the design of a large space structure.

2.2.3 Accelerometers.

Mounted to the FTE directly below, and aligned with the two electromagnetic shakers were two single axis Sunstrand Q-Flex® accelerometers. These accelerometers offer single axis measurement due to its precision made quartz components (28). Figure 2.4 shows the accelerometer placement on the FTE and their proximity to the shakers.

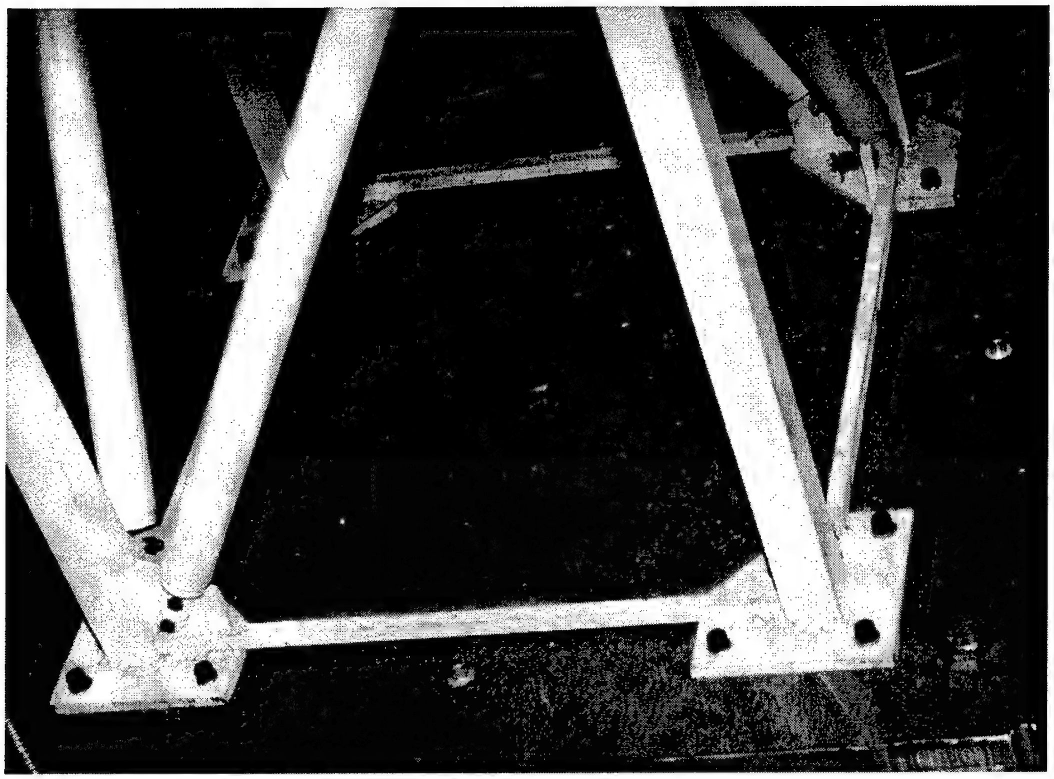


Figure 2.2 FTE Aluminum Base Plate

2.2.4 Scientific-Atlanta Pro Series Dynamic Signal Analyzer.

The Scientific-Atlanta Pro Series Dynamic Signal Analyzer, SA390, is an easily configurable eight channel analyzer that allows for live acquisition, analysis, and display of waveform signals. The SA390 generated the electromagnetic excitation signals, and it acquired both accelerometer outputs.

The SA390, in conjunction with a power amplifier, was configured to provide continuous white noise to both shakers at a setting of 2220 mV (root mean squared) amplitude. Only frequencies below 100 Hz are of interest; therefore, the accelerometer sampling frequency was set to 256 Hz to prevent aliasing. Inman (14) describes aliasing to occur when the sampling rate is set too low to capture the details of the analog signal, and the resulting digital representation will cause high frequencies to appear as low frequencies. In order to recover a signal from its digital samples, the signal must be sampled at least twice the highest frequency in the signal, and experience dictates a sampling rate 2.5 times the

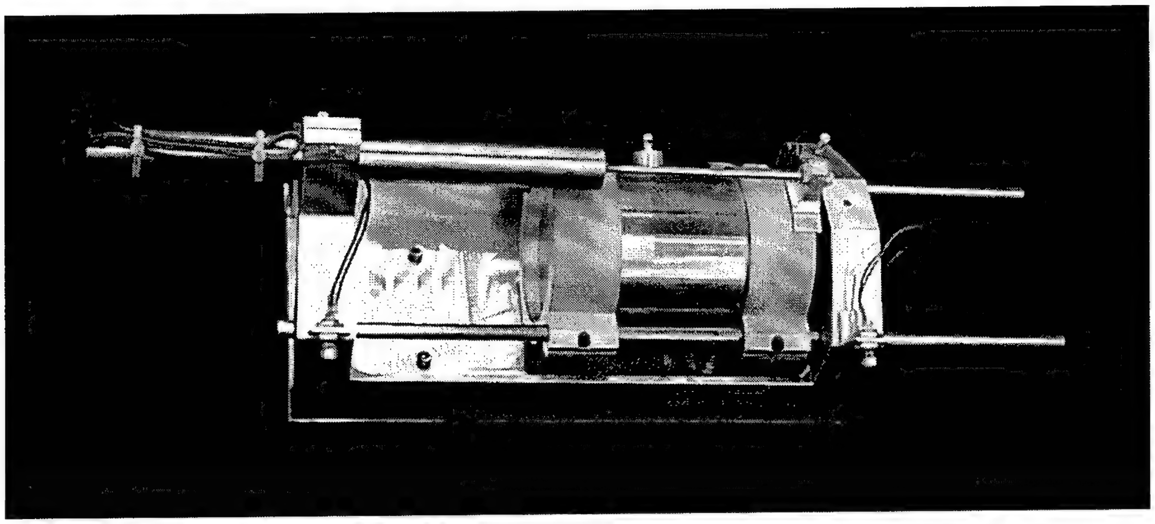


Figure 2.3 FTE Electromagnetic Shaker

highest frequency is optimal (14). To further eliminate aliasing, a low-pass, 100 Hz filter was applied to the signal before it was sent to the shakers.

In addition to aliasing, the SA390 was configured to reduce possible frequency leakage of the accelerometers. Inman (14) describes leakage as the phenomenon that occurs when the signal is cut off at any integral multiple of its period. For signals that contain many different frequencies, this is a problem. This may result in signals being cut-off mid-period. Erroneous frequencies would then appear in the digital representation, because the Digital Fourier Transform of the finite-length signal assumes the signal is periodic (14). The net result is that the actual frequencies leak into a number of fictitious ones. This leakage phenomenon can be improved via windowing the original time domain signal. The time domain signal is multiplied by a window function, which forces the signal to be zero outside the sampling period (14). A Hanning window was used with 90% overlap to reduce leakage. This configuration produced good FRFs; if good results were not obtained, the overlap could have been reduced to yield better FRFs. However, a significant increase in computational time would have been realized with a reduction in overlap. More data would be discarded in the overlapping process. Therefore, additional data would have to be gathered, thus increasing acquisition time.

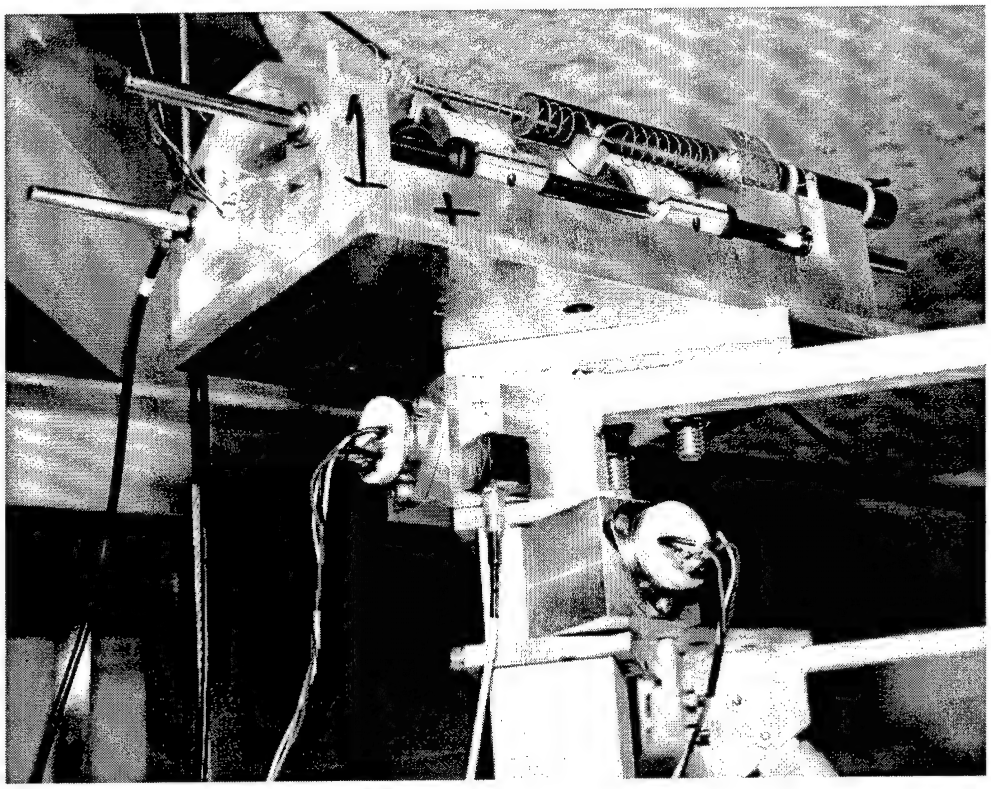


Figure 2.4 Shakers and Accelerometer

To incorporate overlapping, the SA390 was configured to perform a new analysis on a segment of data in which only a portion of the signals has been updated (some old data, some new data). It took from four to seven seconds to gather 400 discrete, FRF data points from 1024 time domain data samples. Another FRF was computed and averaged in with the first via overlapping. This significantly reduced the time needed to compute 100 averages. The overlapping process allowed 36 new data points to be added and 36 of the oldest data points to be dropped from the previous sample for the next FRF computation (29). This resulted in 100 averages being computed in 18 seconds.

The output of the two accelerometers was conditioned and fed into the SA390 where it could be displayed. The SA390 was configured so that the Frequency Response Functions, accelerometer output voltage divided by the excitation force voltage, was displayed. The electromagnetic shakers provided approximately 0.2 pounds of force per applied volt (12). The time domain signals were converted into the frequency domain via the Fast Fourier Transform. Figure 2.5 illustrates the setup.

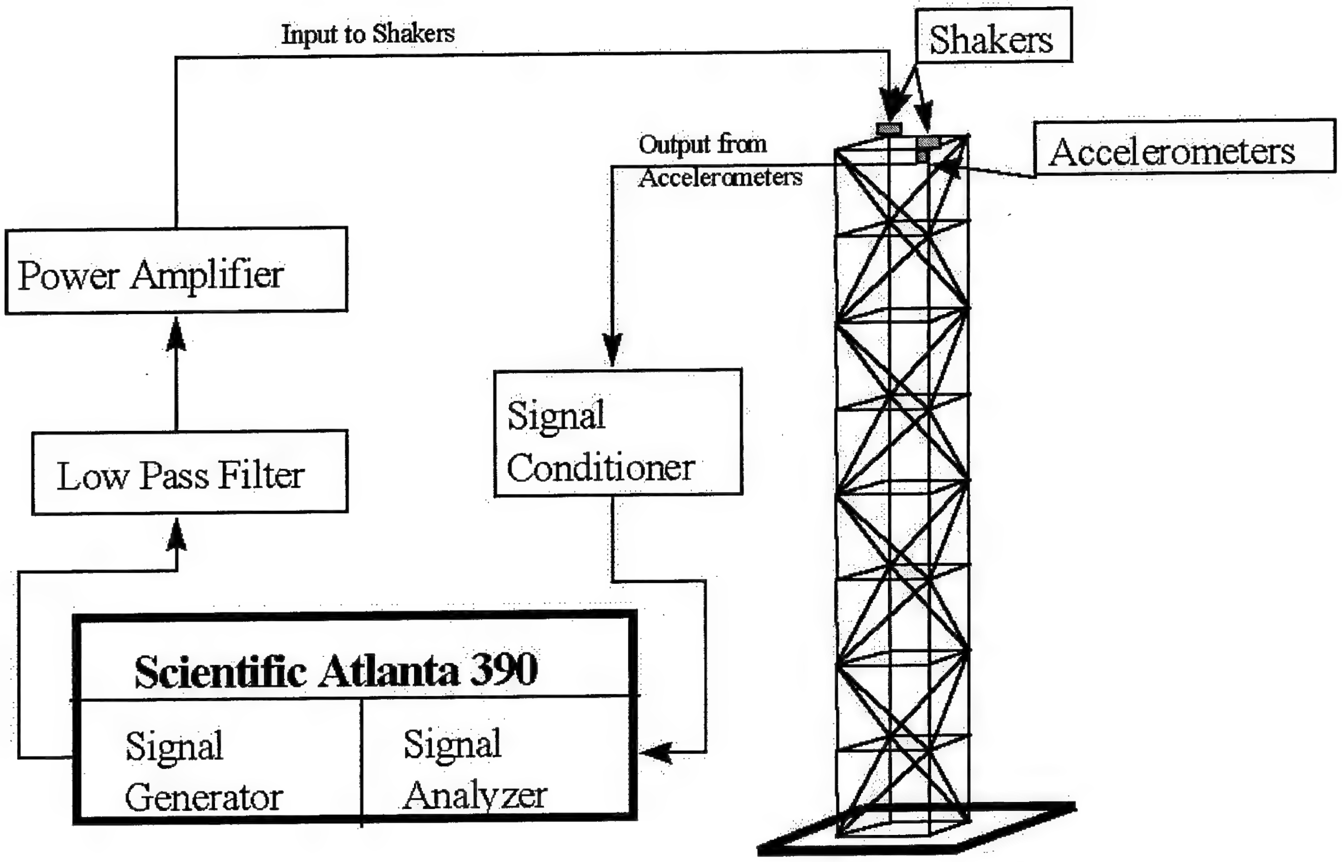


Figure 2.5 Overall System Setup (29)

2.3 Data Acquisition

The SA390 was used to drive the electromagnetic shakers and gather accelerometer data. The SA390 is a versatile, 486 based PC operating on Microsoft® Windows^{TM} 3.1. The SA390 is shown in the bottom right of Figure 2.6 along with other data acquisition hardware.

The SA390 was configured in such a way as to allow Matlab^{TM} access and control of the SA390. A Matlab^{TM} m-file was created that told the SA390 when to begin acquiring data and to save the data in a specified format that could be easily read by Matlab^{TM} for further analysis. Utilizing the Dynamic Data Exchange capability between the SA390's data acquisition software and Matlab^{TM} facilitated the gathering of data.

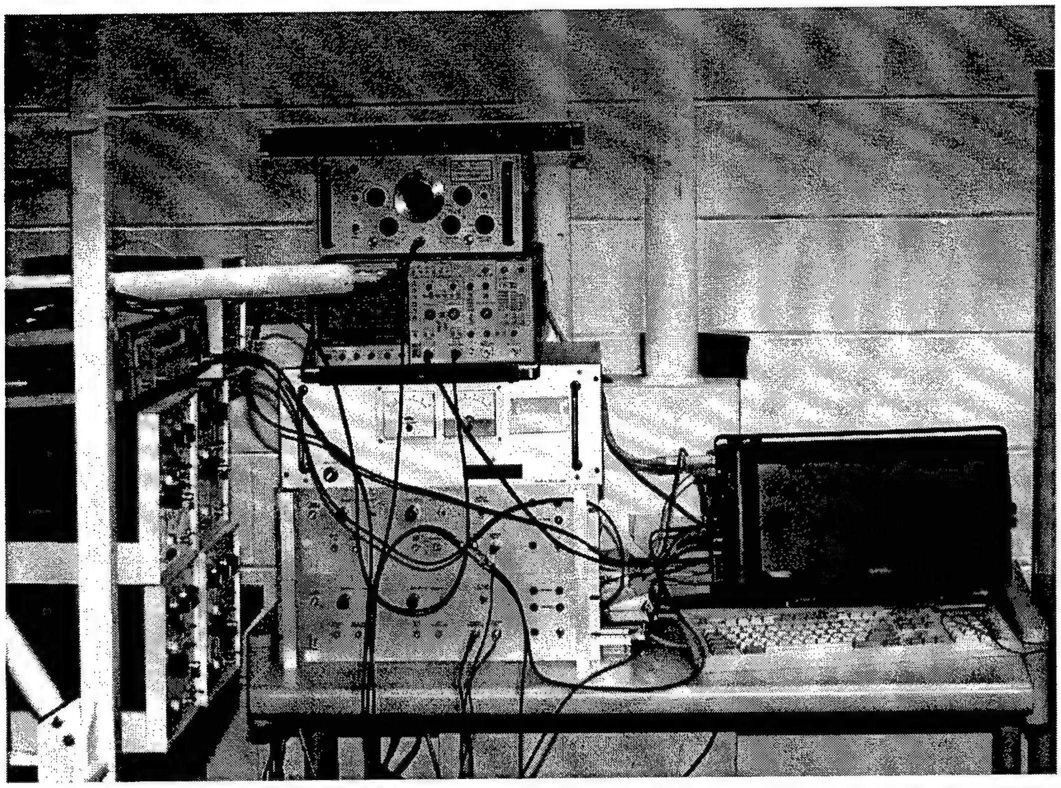


Figure 2.6 Experiment Data Acquisition Setup

One FRF from 0-100 Hz took approximately four to seven seconds to measure. One hundred FRF's were gathered for each damage case, of which there were 112, plus one undamaged case for a total of 113 different cases. There were 14 damage cases simulated for each bay. It took approximately 15 minutes to gather 100 FRFs for one damage class. Four hundred frequency data points were computed from 1,024 time domain samples via the Fast Fourier Transform described previously. This reduced the probability of leakage and aliasing.

Exactly 22,600 FRF's were gathered from the FTE in various states of simulated damage. Damage was simulated in the FTE in the following ways:

- Complete removal of each diagonal
- Removal of each diagonal & replacement with a partially damaged diagonal
- Complete removal of all two diagonal combinations in each bay

Figure 2.7 and Figure 2.8 show an undamaged and damaged diagonal. As can be seen, the partially damaged diagonal is approximately a 50% reduction in cross-sectional area. The latter two damaged cases listed above were not addressed by Swenson (29). The other two damage cases were added, because it was felt these are also very likely damage scenarios, especially in the case of partial damage.

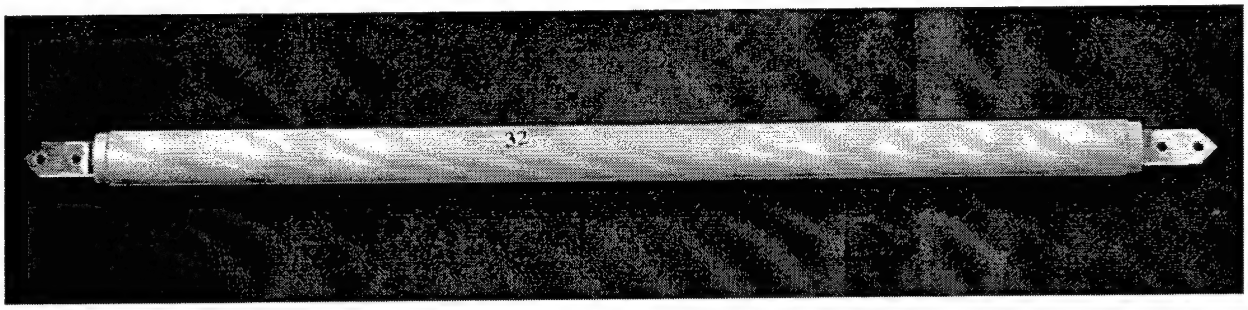


Figure 2.7 Undamaged FTE diagonal

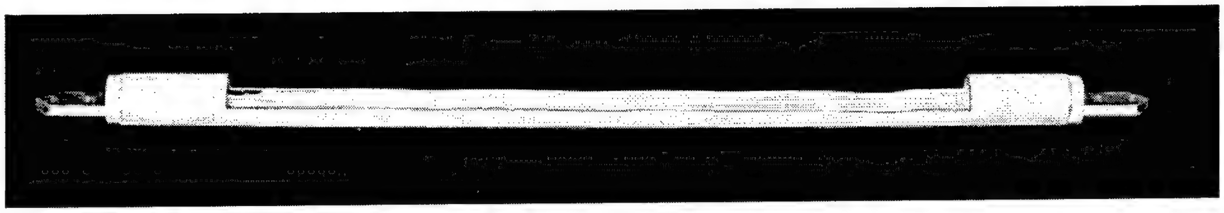


Figure 2.8 Partially Damaged FTE Diagonal

After the 22,600 FRF's were gathered, they were used to train pattern classifiers. These pattern classifiers would then be utilized to identify damage in the FTE given an FRF of the FTE from some unknown damaged or undamaged state. The pattern classifier development and theory will be discussed in Chapter 3, and the results of testing the pattern classifiers will be presented in Chapter 4.

III. Damage Detection Development

3.1 Introduction

Two different damage detection methodologies will be presented in this chapter. The first method relies on a data reduction and feature technique commonly used in the field of speech recognition. The second method extracts resonant and/or anti-resonant frequencies of the FTE in different states of damage. Although they are considerably different, both methodologies rely on the same FRF data for all of the damage cases and for training of the pattern classifiers, or damage detection algorithms.

3.2 Speech Spectra Method

The speech spectra method developed is a data reduction technique that is a derivative of the isolated word recognition problem based upon Hidden Markov Model (HMM) theory. Rabiner, et. al (24) define HMM as a doubly stochastic process with an underlying process that is not directly observable, but, instead, it can only be observed through another set of stochastic processes that produces the sequence of observed symbols.

A front-end signal processing tool is generally needed in speech recognition. Two, typical processing tools, identified by Rabiner, et. al (25), are the bank-of-filters model and a linear predictive coding model. The bank-of-filters signal processing tool will be the basis for the damage identification in this work. Instead of looking at frequencies of 100-3000 Hz typical for telephone quality signals (25), frequencies below 100 Hz will be used in this work in analyzing the FTE.

The usual choice in filter banks for speech recognition is the uniform filter bank (25). However, Colombi (6) showed that non-uniform or nonlinear filter banks empirically improve speech recognition performance. Davis, et.al (8) stated that since there is a known variation of the ear's critical bandwidth with frequency, filters are spaced linearly at low frequencies and logarithmically at higher frequencies. On a mel-frequency scale, the filters are spaced linearly according to Equation 3.1. The mel-frequency scale is based upon the critical band scale associated with speech (25). The spacing of the filters along the mel-frequency scale is based on perceptual studies, which is intended to select frequency bands that give equal contribution to speech articulation (25). In addition, studies have shown that human frequency perception of sounds does not follow a linear scale (25).

$$Mel(f) = 2595 log_{10}(1 + \frac{f}{50}) \tag{3.1}$$

$$y(n) = 50(10^{n*s/2595} - 1) \tag{3.2}$$

where

$$f = \text{discrete frequency in Hz}$$

$$50 = \text{linear filter cut-off, i.e., 50 Hz on a normal scale}$$

$$y(n) = \text{center frequencies for each n filter on analog scale}$$

$$n = \text{1,2,...,n (filter index, 20 filters in this application)}$$

$$s = \text{spacing of filters on mel-scale (50 mels)}$$

The filters used in this work were linearly spaced up to 50 Hz and logarithmically spaced from 50 to 100 Hz on the analog frequency scale. This filtering scheme was used for two reasons. First, linear filter spacing at lower frequencies and logarithmic after, proved successful in speech recognition research by Colombi (6), with the obvious exception that frequencies far greater than 100 Hz were of interest in speech identification. More importantly, it was felt that more noise was being realized at higher frequencies than at lower ones in the FTE. A linear filter spacing at lower frequencies would provide more filters at lower frequencies, thereby resulting in better resolution there than at higher frequencies. It was felt this configuration would lead to better damage detection results.

A linear weighting function was created for each filter, because the filters chosen were triangular. After determining the number of filters (which will be discussed later), the analog frequency limits, i.e., the lower, center, and upper analog frequencies of each linear filter, were determined via Equation 3.2, given a filter spacing of 50 on the mel-frequency scale. The center frequency for the first filter was 50 mels, and all subsequent center frequencies were 50 mels greater than the previous center frequency. Equation 3.2 converts the center frequencies on the mel-scale back to the analog scale. The lower and upper frequencies for the first filter on the mel-scale were 0 and 100, respectively. Similar to the center frequencies, all subsequent lower and upper frequencies were 50 mels greater than the previous lower and upper frequencies. The number of weights per filter equaled the number of discrete FRF data points. On the mel-frequency scale, the area under each filter is unity. The filter weights were then multiplied by the raw FRF frequency domain data gathered by the SA390 via a fast fourier transform of the time domain data. A total of 20 triangles or filters was used for each of the two accelerometer FRFs. Swenson (29) had used 10 filters, but early attempts to duplicate the results obtained by Swenson were unsuccessful. In addition, this work attempted to create a damage identifier capable of detecting more damage cases, including partially damaged member configurations. For these reasons, it was felt that 20 filters would increase the resolution and capability of the damage identifier. This multiplication of the FRF data and the filters represents a discrete integration. The output of these filters are referred to as Mel-Frequency Spectral Coefficients (MFSC) in the field of speech analysis. The filters used in this thesis can be seen in Figure 3.1.

With these filters, two, 400 data point FRFs are reduced to two, 20-point vectors, a reduction of 95% resulting in a 1-by-40 size vector. This has the effect of reducing both the amount of data and computation time associated with the pattern classifier.

Once this data reduction technique had been accomplished with the triangular filters, a Discrete Cosine Transform (DCT) was applied to the MFSCs. The DCT is a common tool used in speech signal processing for data reduction. Jain (15) defines the N-by-N DCT per Equations 3.3 and 3.4 for discrete, integer k. In our application, N is the length of the FRF data, which is 400 discrete, data points.

$$c(k,n) = \frac{1}{\sqrt{N}} \tag{3.3}$$

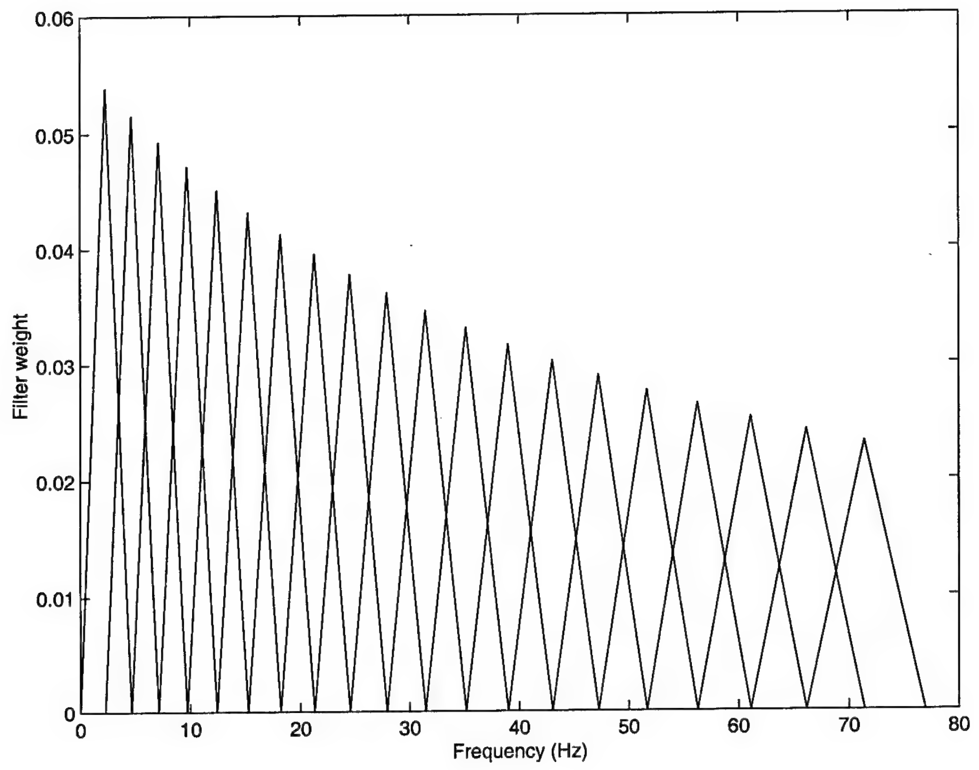


Figure 3.1 Triangular Filters

for

$$k = 0,\ 0 \leq n \leq N-1$$

$$c(k,n) = \sqrt{\frac{2}{N}} \cos \frac{\pi(2n+1)k}{2N} \tag{3.4}$$

for

$$1 \leq k \leq N-1,\ 0 \leq n \leq N-1$$

Some properties of the DCT are as follows (15):

- The cosine transform, $\mathbf{C}$ (matrix of $c(k,n)$), is real and orthogonal:

$$\mathbf{C}^{-1} = \mathbf{C}^T \tag{3.5}$$

- The cosine transform is a fast transform.
- The cosine transform has excellent energy compaction for highly correlated data.

Equation 3.6 illustrates the DCT used in this work (6), which is different than the DCT described by Jain (15).

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j-0.5)\right) \tag{3.6}$$

where

$$\begin{aligned} m_j &= \text{Mel-Frequency Spectral Coefficients (MFSCs)} \\ N &= \text{Number of MFSCs} \end{aligned}$$

The MFSCs are referred to as Mel-Frequency Cepstral Coefficients (MFCC), once the DCT is applied. As stated previously, the MFSCs are the product of the raw FRF data and the linear weights. The DCT both decorrelates and orthogonalizes the data. The above process is illustrated in Figure 3.2, starting with the time domain data and ending with the MFCCs.

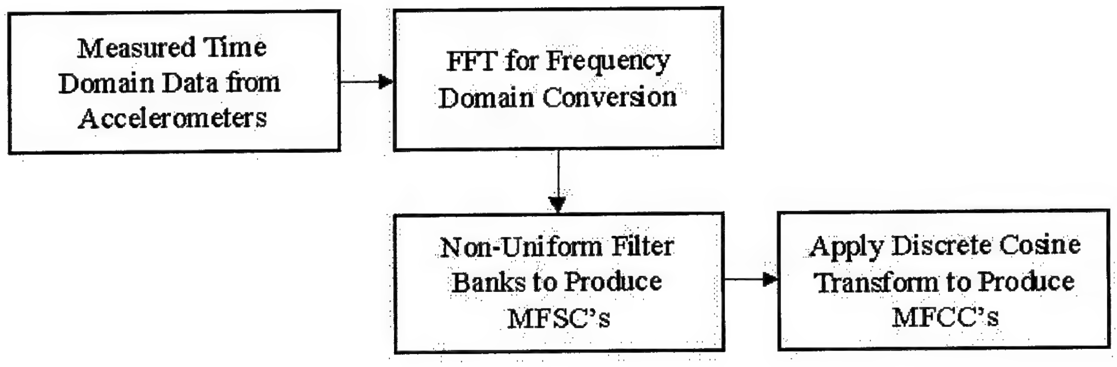

Measured Time Domain Data from Accelerometers
FFT for Frequency Domain Conversion
Non-Uniform Filter Banks to Produce MFSC's
Apply Discrete Cosine Transform to Produce MFCC's

Figure 3.2 Data Reduction Process via Speech Spectra Method

The process outlined in Figure 3.2 was used to train the pattern classifiers. Training will be discussed in subsequent sections in this chapter. In addition, this process will be used when reducing the data associated with a unknown or test FTE damage configuration. This test vector, 1-by-40 long, will be compared to all other damage cases to determine the unknown, damaged state in the FTE.

3.3 Damage Detection via Shifts in Resonant and Anti-Resonant Frequencies

The speech spectra method outlined above is a very effective data reduction technique. However, a method utilizing modal parameters, i.e., natural frequencies, mode shapes, etc., was sought since these modal parameters are well understood and are of physical significance in any structure. To illustrate these frequency shifts due to damage, the Eigensystem Realization Algorithm, or ERA, was executed on finite element models of the FTE in the following configurations:

- Undamaged
- Diagonal 1 Removed
- Diagonal 1, 50% damaged

ERA provides considerable insight into some of the modal parameters of the FTE. Following a discussion of ERA and ERA results on the FTE, a description of the resonant and anti-resonant frequencies damage detection algorithm development will be provided in this section. Finally, an example will be provided to illustrate the resonant and anti-resonant frequency algorithm. As will be seen, this algorithm development required several iterations before suitable parameter extraction and case comparison techniques were found.

3.3.1 Eigensystem Realization Algorithm. Before discussing ERA testing and results on the FTE, a brief mathematical description shall be provided. As discussed in Chapter I, natural frequency shifts have been used to determine whether or not damage exists in a structure. ERA will show these shifts in the FTE.

ERA starts with a finite dimensional, discrete-time, linear, time invariant, state-space description (18) as given in the following equations:

$$x(k+1) = Ax(k) + Bu(k) \tag{3.7}$$

$$y(k) = Cx(k) \tag{3.8}$$

where

$$\begin{aligned} x &= \text{an n-dimensional state vector} \\ u &= \text{an m-dimensional control input} \\ y &= \text{a p-dimensional output vector} \\ A, B, C &= \text{the state, contol, and output matrices} \\ k &= \text{the integer sample indicator} \end{aligned}$$

The discrete time, unit-impulse response, Y(k), of the system described in Equations 3.7 and 3.8 is referred to as the Markov parameter (18), as illustrated in the following equation:

$$Y(k) = CA^{k-1}B \tag{3.9}$$

The Markov parameters can then be used to define a larger matrix of measurements, $H_{ij}(k)$, for any i and j. $H_{ij}(k)$ is also known as the Hankel matrix.

$$H_{ij}(k) = \begin{pmatrix} Y(k) & Y(k+1) & \dots & Y(k+j) \\ Y(k+1) & Y(k+2) & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ Y(k+i) & Y(k+i+1) & \dots & Y(k+i+j) \end{pmatrix} \tag{3.10}$$

$$= O_i A^{k-1} R_j \tag{3.11}$$

where

$$\begin{aligned}
O_i &= \text{is the observability matrix} \\
&= \begin{bmatrix} C \\ CA' \\ \cdots \\ \cdots \\ \cdots \\ CA'^i \end{bmatrix} && (3.12) \\
i,j &= \text{number of rows and columns, respectively} \\
R_j &= \text{is the controllability matrix} \\
&= \begin{bmatrix} B' & A'B' & \ldots & A'^j \end{bmatrix} && (3.13) \\
A' &= e^{A\Delta t} && (3.14) \\
B' &= \int_0^{\Delta t} e^{A\sigma} d\sigma B && (3.15)
\end{aligned}$$

ERA is based on a singular value decomposition of the Hankel Matrix. The eigenvalues, λ_i, of the continuous time state matrix, A, are found from the eigenvalues of A' per Equation 3.16.

$$\begin{aligned}
\lambda_i &= \frac{\ln \lambda_i'}{\Delta t} && (3.16) \\
\lambda_i' &= \text{the eigenvalues of } A'
\end{aligned}$$

An ERA Matlab^{TM} m-file created by Mr. Joseph Hollkamp of the Air Force Research Laboratory (AFRL) was used.

ERA was utilized for better modal resolution associated with natural frequency shifts, because these frequency shifts would indirectly form the basis of the proposed damage detection algorithm. The m-file requires the user to input the number of rows and columns used in the Hankel matrix analysis. At a minimum, it is recommended that the number of columns be at least twice the number of modes expected (7). Additionally, more columns should be used if the data is noisy. Typically, it is recommended that the number of rows be two or three times the number of columns to obtain satisfactory results (3). Once the Hankel matrix is formed, the m-file algorithm factorizes it via a singular value decomposition (7).

To illustrate the frequency shifts, ERA analysis was conducted on the FTE in three, different configurations, as discussed previously in this section, using Finite Element Model generated data. Figure 3.3 illustrates the Frequency Response Functions for each of the cases listed above out to 60 Hz. ERA analysis was performed on these FRFs, and the results are illustrated in Tables 3.1, 3.2, and 3.3.

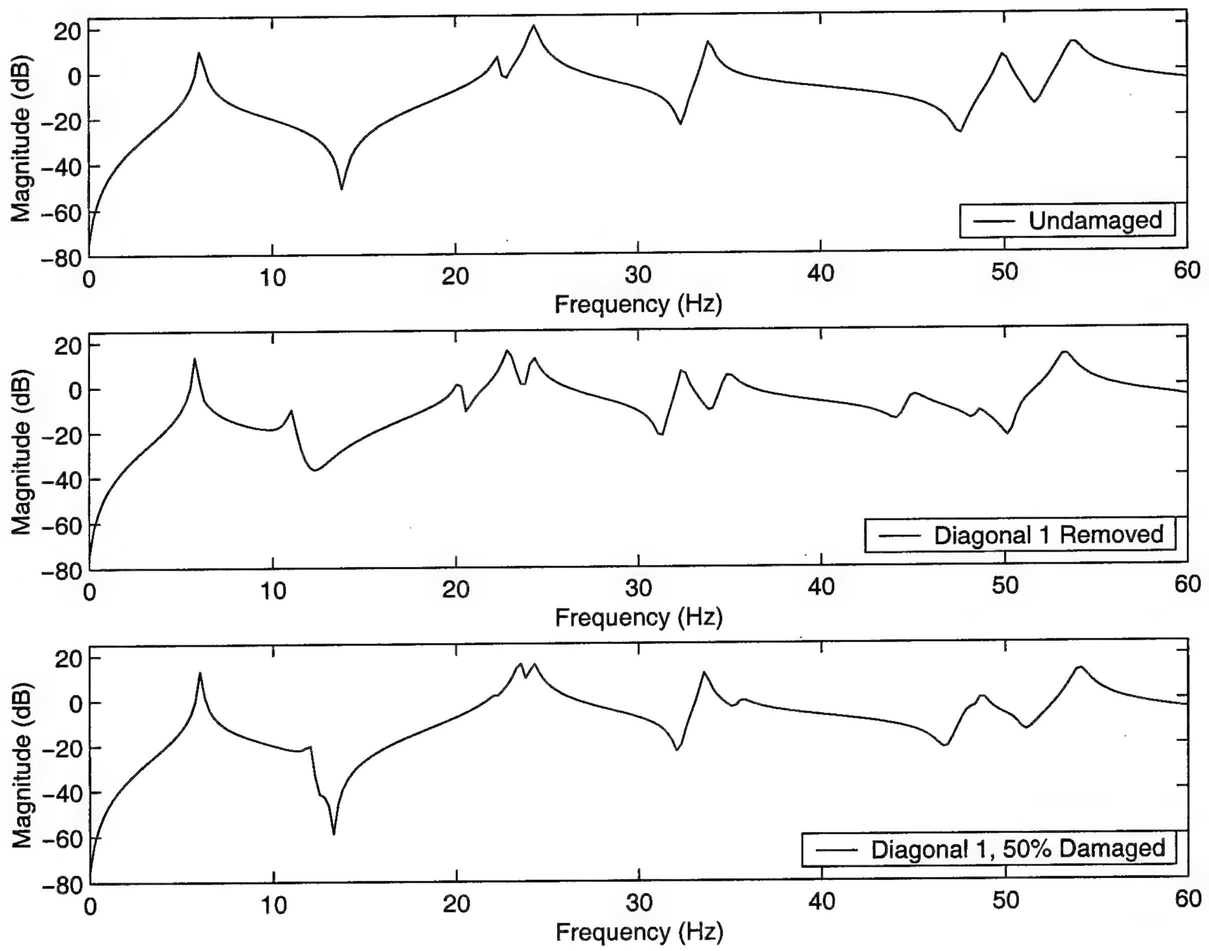


Figure 3.3 FTE Frequency Response Functions, x-Direction Accelerometer

Table 3.1 ERA Results For Undamaged FTE

Natural Frequency	Frequency(Hz)	EMAC(%)
1st	7.80	98.02
2nd	24.65	90.28
3rd	33.89	99.51
4th	49.90	97.45
5th	54.28	92.67

Table 3.2 ERA Results For Diagonal 1 Removed

Natural Frequency	Frequency(Hz)	EMAC(%)
1st	7.41	99.84
2nd	11.08	94.57
3rd	20.08	78.94
4th	23.11	85.18
5th	32.68	92.67
6th	35.08	90.63
7th	53.18	97.94

Tables 3.1, 3.2, and 3.3 illustrate the shifts in frequency in addition to different numbers of natural frequencies from one damage state to the next. We conjecture that the modes measured from the y-direction accelerometer are manifested more in the x-direction in some damage cases than others. This validates the assumption of the uniqueness of differing damage states, which is the basis for damage detection. Extended Modal Assurance Criterion (EMAC) is a measure of the degree of confidence that the predicted natural frequency is actually a natural frequency, or a mode. If ERA produces a frequency with an EMAC greater than 85%, we can safely say that the frequency is a resonant frequency, or a mode (3). The only evident drawback in this analysis is the inherently lower EMAC values associated with closely spaced modes, as can be seen with Diagonal 1 removed. Also of interest is ERA's inability to detect the apparent 22 Hz resonant frequency associated with the undamaged configuration. Several different size configurations of the Hankel matrix, i.e., number of rows and columns, were unsuccessfully attempted to capture this resonant frequency. Poor ERA resolution, and close proximity to the 24.5 Hz resonant frequency probably contributed to this resonant frequency not being detected.

Table 3.3 ERA Results For Diagonal 1 50% Damaged

Natural Frequency	Frequency(Hz)	EMAC(%)
1^{st}	7.75	99.89
2^{nd}	23.87	85.35
3^{rd}	33.61	96.58
4^{th}	48.73	93.65
5^{th}	53.88	97.79

3.3.2 Resonant and Anti-Resonant Damage Detection Algorithm Development. Once it was shown via ERA the extent to which the natural frequencies shifted in different damage cases, a Matlab^{TM} m-file was written that extracted the resonant frequencies, which correspond to the peaks in the FRF plots. In addition to extracting the natural frequencies, the m-file extracted the local peaks in the FRF magnitude spectra plots that are not necessarily natural frequencies. In most cases, these local peaks are modes associated with the other orthogonal Degrees of Freedom (DOFs) that crept into the sensor response. It was not possible to configure the accelerometers such that a mode would only be seen in one accelerometer and not the other accelerometer. Therefore, these smaller peaks at a particular frequency would be evident in the FRF of a DOF that should not respond to that mode. This additional information was used to retain saliency of the data for each of the damage cases. These smaller peaks could be unique to a specific damage case. The algorithm was also capable of extracting the anti-resonant frequencies and local minima. Damage detection results using resonant frequencies only will be compared to those using both resonant and anti-resonant frequencies in Chapter IV.

The first step in the creation of this algorithm was to define frequency ranges in which the resonant frequencies could be stored in a consistent manner from one damaged state to the next, so they could be compared for damage detection. This was an iterative task, because some of the resonant frequencies were closely spaced. The FRF magnitude spectra was analyzed in 11 data point intervals, starting from 4 Hz and ending at 54 Hz. This corresponds to 200 discrete data points with the magnitude being known at every 0.25 Hz increment. The maximum value was found in that 11 point, 2.50 Hz interval. For example, the algorithm would analyze magnitudes from 4.00 Hz to 6.50 Hz. The maximum value in that range would be compared to the maximum magnitude value from 4.25 Hz to 6.75 Hz. If the maximum value from 4.00 Hz to 6.50 Hz was less than the maximum value from 4.25 Hz to 6.75 Hz, it was discarded. If not, that maximum value was retained and repeatedly compared to subsequent 11 point intervals. That maximum value would then be classified a resonant frequency if it was still a maximum value when the 11 point magnitude window moved past that particular magnitude data point.

An identical process was accomplished to find the anti-resonant frequencies with the obvious difference being that instead of analyzing the maximum values, the minimum values were found.

Once all the resonant and anti-resonant frequencies were found, a method to organize these frequencies in a consistent manner was needed. This was necessary due to there being unequal numbers of resonant and/or anti-resonant frequencies from one damaged state to the next, as illustrated by ERA. By some trial and error, the resonant and anti-resonant frequencies were arranged in 2 Hz intervals. For example, if there was a resonant frequency greater than or equal to 4 Hz and less than 6 Hz, it would be stored in the first element in a vector comprised of resonant and anti-resonant frequencies. If there was not a resonant frequency greater than or equal to 4 Hz and less than 6 Hz, a zero would be stored in the first element of the vector. It turned out that, in all the damage cases, consecutive resonant and anti-resonant frequencies were never closer than 2 Hz apart. Coupling this with the 11 point intervals described above produced the best results on the measured FTE data. Anything less than 11 data point intervals produced spurious resonant and anti-resonant frequencies. Any future attempt to utilize this method on a different structure would probably require a different customization, and no method for automating this process was attempted. Figure 3.4 illustrates this process.

An example follows that is intended to provide the reader with a better understanding of the damage detection algorithm discussed previously. We begin with the average of 10, 400 data point FRFs for both the x and y direction accelerometers, shown in Figure 3.5.

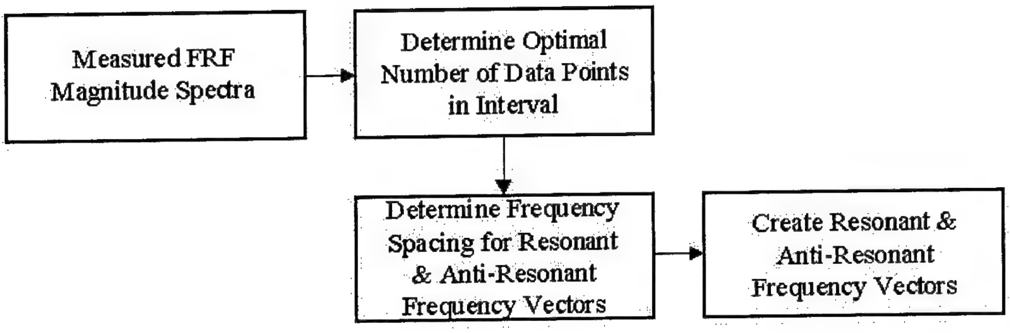

Measured FRF Magnitude Spectra
Determine Optimal Number of Data Points in Interval
Determine Frequency Spacing for Resonant & Anti-Resonant Frequency Vectors
Create Resonant & Anti-Resonant Frequency Vectors

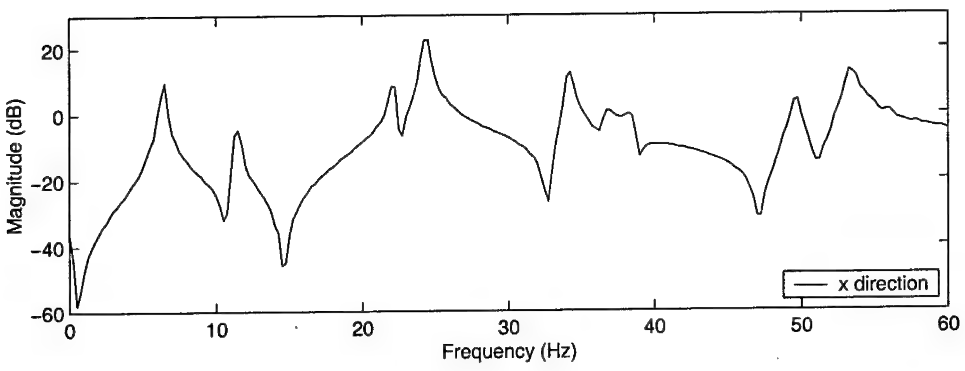
Figure 3.4 Resonant & Anti-Resonant Frequencies Algorithm Development

As seen in Figure 3.5, there is a clearly defined resonant frequency at approximately 6.5 Hz (x-accelerometer). This example will show how the algorithm proceeded to determine that 6.5 Hz was a resonant frequency. As mentioned previously, frequencies will only be analyzed from 4 to 54 Hz, in 11 discrete data point intervals. Analyzing data from 4 to 54 Hz, instead of 0 to 100 Hz, halved the amount of data to be analyzed. Starting at 4 Hz, and ending at 9.0 Hz, Table 3.4 illustrates the successive intervals used in the analysis of the 6.5 Hz resonant frequency.

As Table 3.4 indicates, the magnitude of the resonant frequency at 6.5 Hz was greatest throughout intervals 1 through 11. If the magnitude at 9.0 Hz had been greater than 9.7 dB, then the algorithm would simply discard the resonant frequency at 6.5 Hz and retain the one at 9.0 Hz. All 11 data point intervals following 9.0 Hz would then be analyzed in an identical fashion. This procedure was continued to determine the remainder of the resonant & anti-resonant frequencies in both the x and y direction FRFs.

3.4 Damage Detection Training

Pattern Training is defined by Rabiner (25) to occur when one or more test patterns corresponding to speech sounds of the same class are used to create a pattern representative of the features of that class. The resulting pattern is typically called a reference pattern. The pattern training accomplished in this work is an adaptation of Rabiner's method, since we are interested in the FRF spectra rather than those of speech.



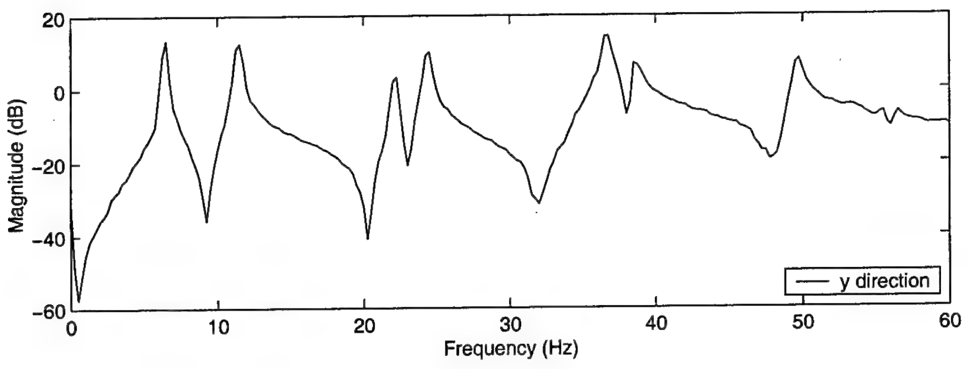


Figure 3.5 Accelerometer FRFs for Diagonal 16 Removed

Training was accomplished in this work with 100 FRFs per damage class. This was done slightly different between the speech spectra method, and the resonant and anti-resonant frequency method.

Training the speech spectra algorithm was accomplished in the following, easily coded manner. First, all of the 100 FRFs for a particular damage class were reduced via the method outlined in Figure 3.2. Second, a matrix of means was calculated from the resulting orthogonal MFCCs, one mean or average per damage class. Finally, the MFCCs were used to compute covariances for each of the damage class vectors. These covariances, which are the square of the standard deviations, are then used to find the standard deviations. Two matrices resulted from this training. One matrix contained all the averaged MFCCs, with each column representing a different damage case. The second matrix contained all the averaged covariances, arranged with each column representing a different damage case. Figure 3.6 illustrates this training process.

Table 3.4 Algorithm Determination of 6.5 Hz Resonant Frequency

Interval	*Frequency (Hz) and Magnitude (dB) Values*										
1^{st}(Hz)	4.0	4.25	4.5	4.75	5.0	5.25	5.5	5.75	6.0	6.25	**6.50**
1^{st}(dB)	-23.4	-21.6	-20.0	-18.4	-15.8	-12.8	-9.7	-7.2	0.0	5.4	**9.7**
2^{nd}(Hz)	4.25	4.5	4.75	5.0	5.25	5.5	5.75	6.0	6.25	**6.50**	6.75
2^{nd}(dB)	-21.6	-20.0	-18.4	-15.8	-12.8	-9.7	-7.2	0.0	5.4	**9.7**	0.4
3^{rd}(Hz)	4.5	4.75	5.0	5.25	5.5	5.75	6.0	6.25	**6.50**	6.75	7.0
3^{rd}(dB)	-20.0	-18.4	-15.8	-12.8	-9.7	-7.2	0.0	5.4	**9.7**	0.4	-5.9
4^{th}(Hz)	4.75	5.0	5.25	5.5	5.75	6.0	6.25	**6.50**	6.75	7.0	7.25
4^{th}(dB)	-18.4	-15.8	-12.8	-9.7	-7.2	0.0	5.4	**9.7**	0.4	-5.9	-8.1
5^{th}(Hz)	5.0	5.25	5.5	5.75	6.0	6.25	**6.50**	6.75	7.0	7.25	7.5
5^{th}(dB)	-15.8	-12.8	-9.7	-7.2	0.0	5.4	**9.7**	0.4	-5.9	-8.1	-11.0
6^{th}(Hz)	5.25	5.5	5.75	6.0	6.25	**6.50**	6.75	7.0	7.25	7.5	7.75
6^{th}(dB)	-12.8	-9.7	-7.2	0.0	5.4	**9.7**	0.4	-5.9	-8.1	-11.0	-12.7
7^{th}(Hz)	5.5	5.75	6.0	6.25	**6.50**	6.75	7.0	7.25	7.5	7.75	8.0
7^{th}(dB)	-9.7	-7.2	0.0	5.4	**9.7**	0.4	-5.9	-8.1	-11.0	-12.7	-13.9
8^{th}(Hz)	5.75	6.0	6.25	**6.50**	6.75	7.0	7.25	7.5	7.75	8.0	8.25
8^{th}(dB)	-7.2	0.0	5.4	**9.7**	0.4	-5.9	-8.1	-11.0	-12.7	-13.9	-15.6
9^{th}(Hz)	6.0	6.25	**6.50**	6.75	7.0	7.25	7.5	7.75	8.0	8.25	8.5
9^{th}(dB)	0.0	5.4	**9.7**	0.4	-5.9	-8.1	-11.0	-12.7	-13.9	-15.6	-16.8
10^{th}(Hz)	6.25	**6.50**	6.75	7.0	7.25	7.5	7.75	8.0	8.25	8.5	8.75
10^{th}(dB)	5.4	**9.7**	0.4	-5.9	-8.1	-11.0	-12.7	-13.9	-15.6	-16.8	-17.9
11^{th}(Hz)	**6.50**	6.75	7.0	7.25	7.5	7.75	8.0	8.25	8.5	8.75	9.0
11^{th}(dB)	**9.7**	0.4	-5.9	-8.1	-11.0	-12.7	-13.9	-15.6	-16.8	-17.9	-18.8

Training the algorithm based upon the resonant and anti-resonant frequencies was a slight variation of the method just described. Ten FRF magnitude averages were computed from the same 100 FRFs, ten FRFs per average, used in the speech spectra method described above for a damage class. The average of 10 FRFs gave better results than just one, as will be discussed in Chapter IV. Next, the resonant and anti-resonant frequencies were extracted and vectorized from the data according to Figure 3.4 for each of the 10 averages. The 10 resulting vectors composed of resonant and anti-resonant frequencies were then averaged to form a single average vector per damage class. As a result of the averaging, the resulting vector was not necessarily discretized in 0.25 Hz increments. Figure 3.7 illustrates the training for this method.

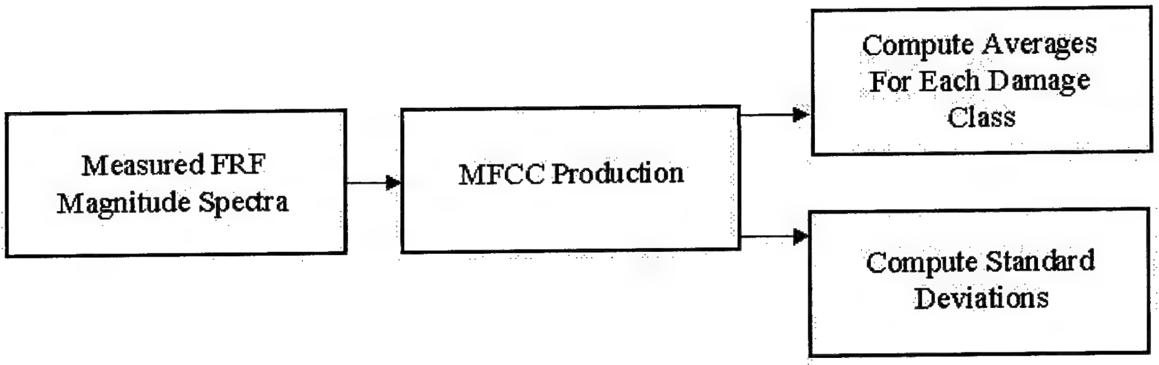

Measured FRF Magnitude Spectra
MFCC Production
Compute Averages For Each Damage Class
Compute Standard Deviations

Figure 3.6 Speech Spectra Method Training Diagram

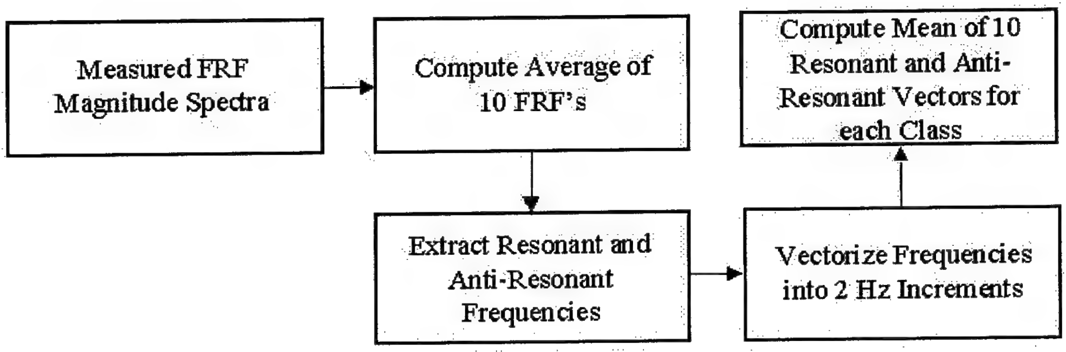

Measured FRF Magnitude Spectra
Compute Average of 10 FRF's
Compute Mean of 10 Resonant and Anti-Resonant Vectors for each Class
Extract Resonant and Anti-Resonant Frequencies
Vectorize Frequencies into 2 Hz Increments

Figure 3.7 Resonant and Anti-Resonant Frequency Method Training Diagram

Covariances were not computed, because, in some cases, a resonant or anti-resonant frequency occurred only once for a damage class despite averaging 10 measured FRFs. This would correspond to a covariance of zero from one data point, which is statistically meaningless.

Continuing with the example of diagonal 16 removed, a portion of the training output is provided in Table 3.5. Only the x-direction resonant frequencies are listed with the first column being the 2 Hz frequency intervals. The x-direction anti-resonant frequencies and the y-direction resonant & anti-resonant frequencies (not listed) are arranged in a similar fashion. Columns 2-11 in Table 3.5 are the x-direction resonant frequencies for 10 averaged FRFs. The final column is the mean of all 10 columns. Columns 2-11 contain the resonant frequencies in 0.25 Hz increments; and, as a result of averaging, the mean of columns 2-11 does not have 0.25 Hz discretization.

Table 3.5 Training Output of Diagonal 16 Removal

Hz	*Training Output, Averages 1-10*										*Mean*
4-6	0	0	0	0	0	0	0	0	0	0	0
6-8	6.50	6.50	6.50	6.50	6.50	6.50	6.50	6.50	6.50	6.50	6.50
8-10	0	0	0	0	0	0	0	0	0	0	0
10-12	11.50	11.50	11.50	11.50	11.50	11.50	11.50	11.50	11.50	11.50	11.50
12-14	0	0	0	0	0	0	0	0	0	0	0
14-16	0	0	0	0	0	0	0	0	0	0	0
16-18	0	0	0	0	0	0	0	0	0	0	0
18-20	0	0	0	0	0	0	0	0	0	0	0
20-22	0	0	0	0	0	0	0	0	0	0	0
22-24	22.00	22.00	22.00	22.00	22.25	22.25	22.25	22.00	22.00	22.00	22.05
24-26	24.50	24.50	24.50	24.50	24.50	24.25	24.25	24.25	24.25	24.50	24.40
26-28	0	0	0	0	0	0	0	0	0	0	0
28-30	0	0	0	0	0	0	0	0	0	0	0
30-32	0	0	0	0	0	0	0	0	0	0	0
32-34	0	0	0	0	0	0	0	0	0	0	0
34-36	34.25	34.25	34.00	34.25	34.25	34.25	34.25	34.25	34.25	34.25	34.23
36-38	36.75	36.75	37.00	36.75	36.75	36.75	37.00	36.75	37.00	36.75	36.83
38-40	38.25	38.25	38.25	38.25	38.50	38.25	38.25	38.25	38.25	38.50	38.30
40-42	0	0	0	0	0	0	0	0	0	0	0
42-44	0	0	0	0	0	0	0	0	0	0	0
44-46	0	0	0	0	0	0	0	0	0	0	0
46-48	0	0	0	0	0	0	0	0	0	0	0
48-50	49.75	49.75	49.75	49.75	49.75	49.75	49.50	49.50	49.75	49.50	49.68
50-52	0	0	0	0	0	0	0	0	0	0	0
52-54	53.25	53.50	53.50	53.50	53.50	53.50	53.25	53.50	53.25	53.50	53.43

3.5 Data Variation

The FRFs for a given damage class turned out to be quite variable. This variability led to the phenomenon discussed above in which a resonant and/or an anti-resonant frequency occurred only one time out of ten in a given 2 Hz frequency range or window. In most cases, when a resonant and/or anti-resonant frequency occurred once, the preceding or subsequent 2 Hz window contained a frequency found for the corresponding resonant or anti-resonant frequency.

Although this variability proved to be somewhat difficult in training the pattern classifiers, it is typical of real structures that are very susceptible to operating conditions. On the other hand, this significant variability in the training data would serve to hopefully accommodate any unknown, random variation sources in testing the pattern classifiers on the FTE using different FRF data than that used for training. It was hoped that an unknown, test FRF would fall into this apparent variability window encountered in training; therefore, the correct damage case would then be identified. Figures 3.8, 3.9, and 3.10 shows this variation at different damage locations on the FTE.

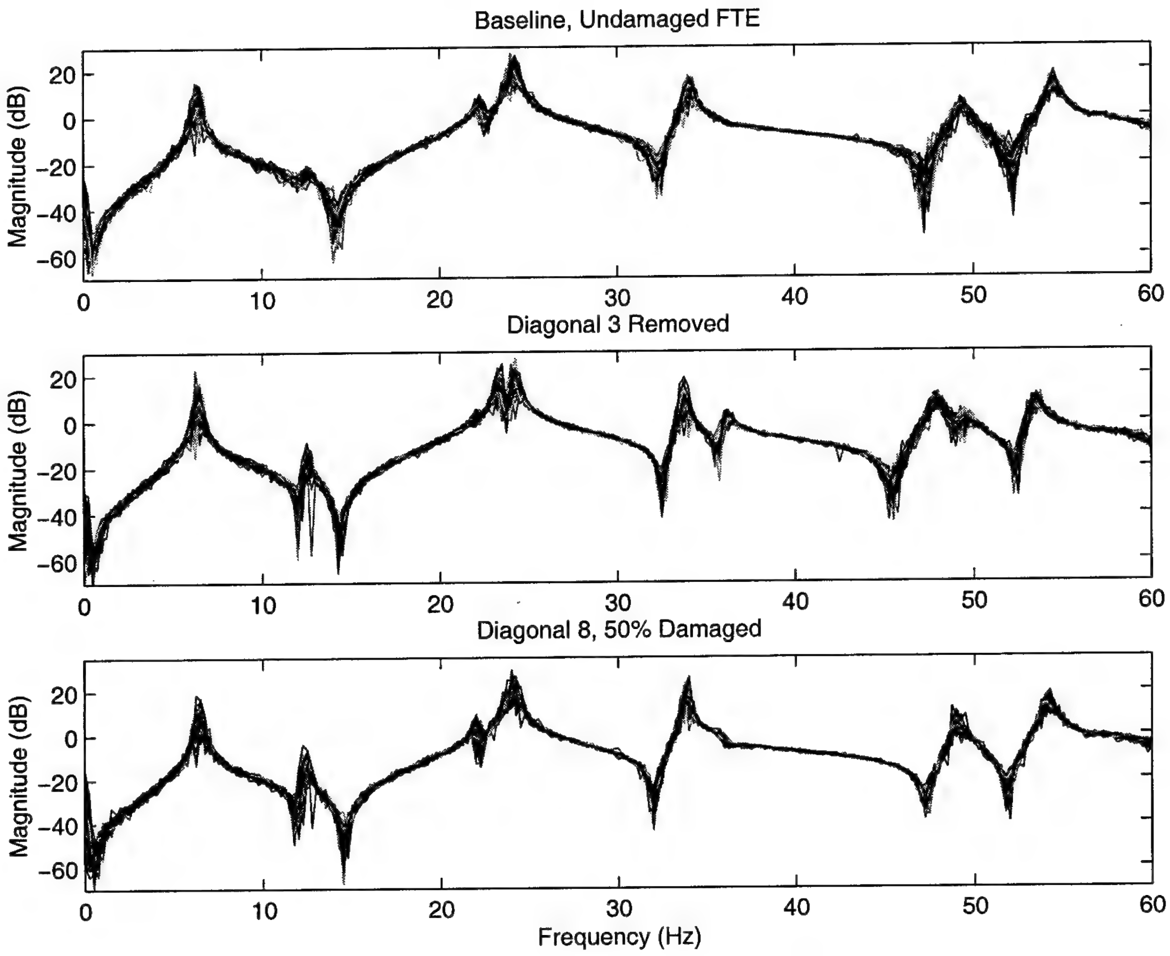


Figure 3.8   FRF Variability in the FTE, Bays 1 and 2

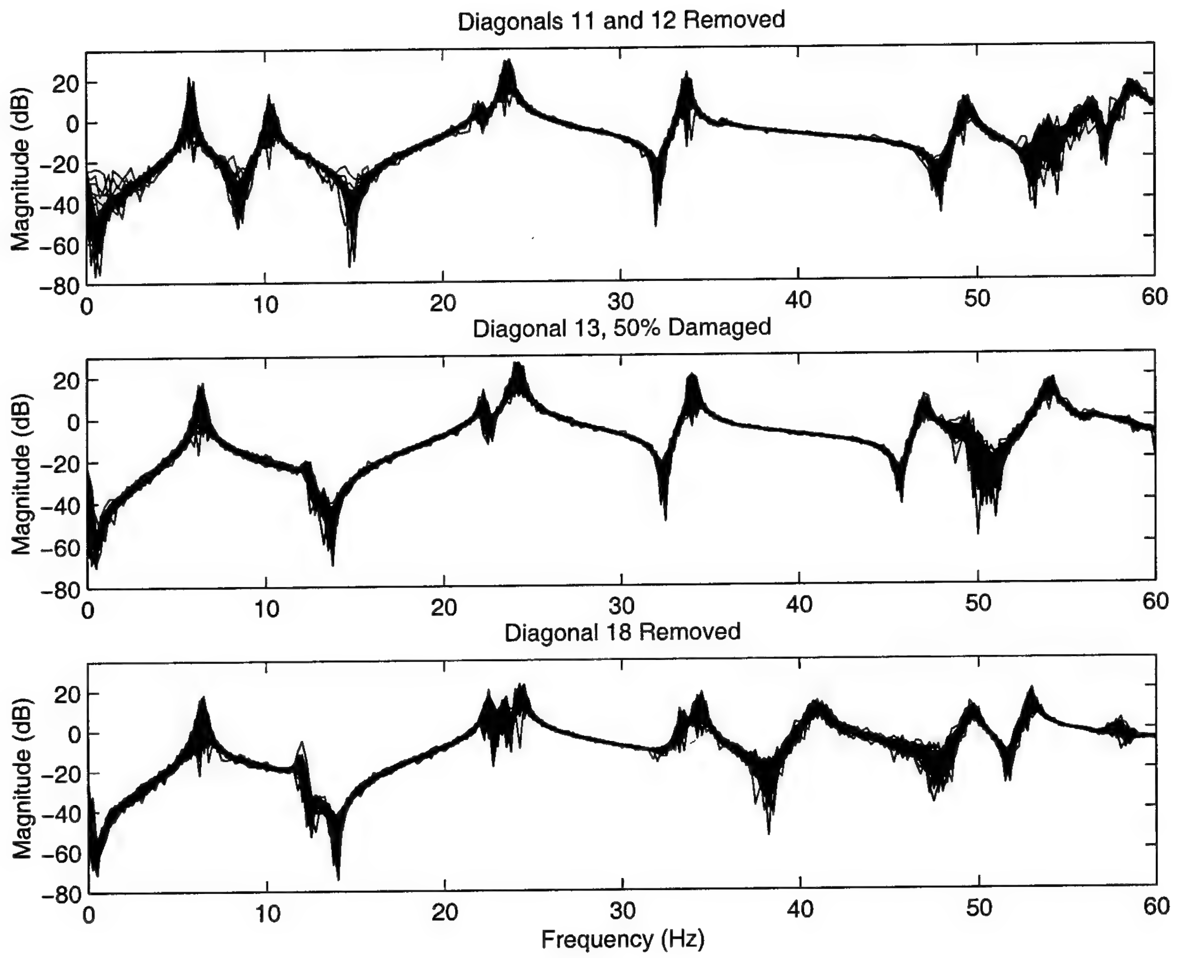


Figure 3.9 FRF Variability in the FTE, Bays 3, 4, and 5

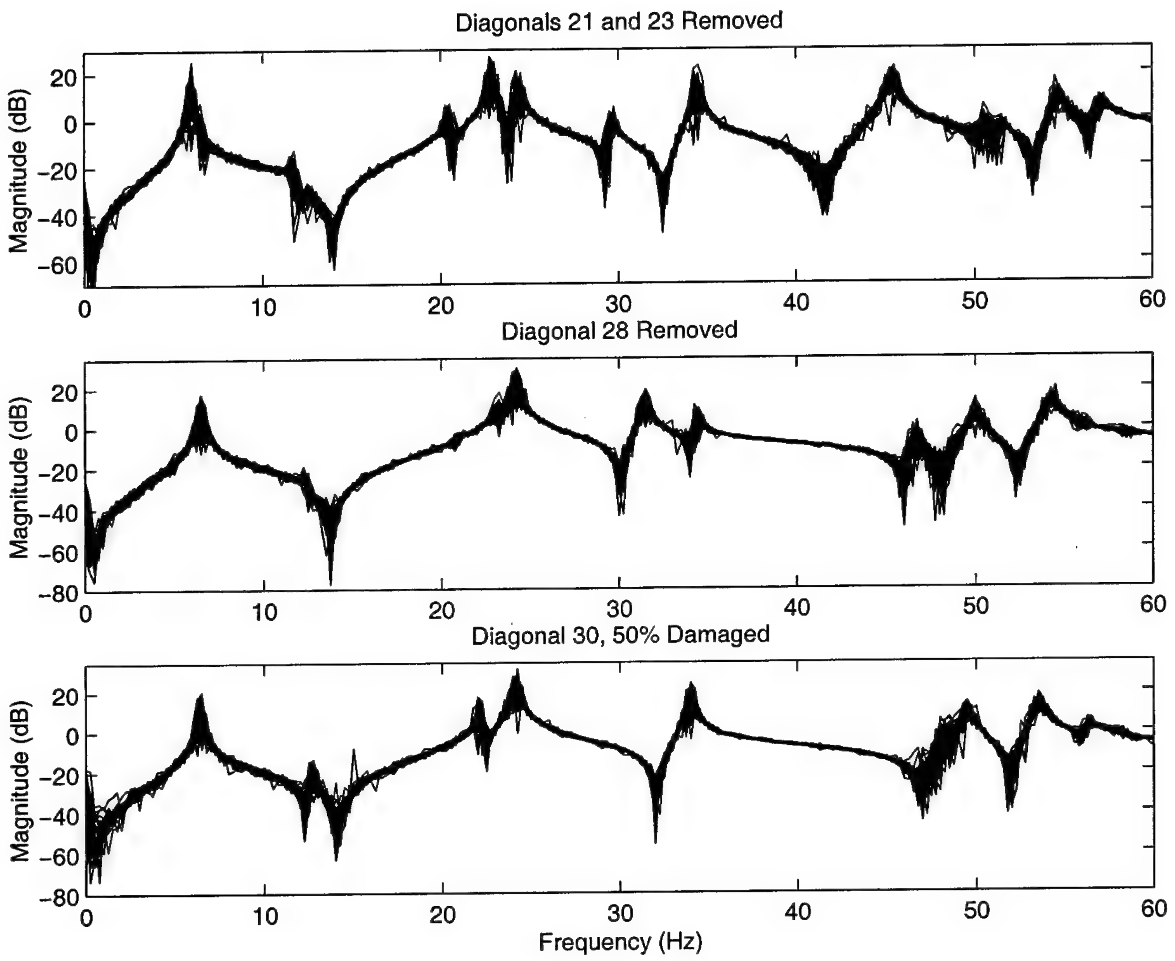


Figure 3.10 FRF Variability in the FTE, Bays 6, 7, and 8

3.6 Pattern Classifiers

One of the pattern classifiers used was adapted from Duda and Hart (9). The Gaussian classifier used in this work is referred to as a Bayes classifier per Duda and Hart. Probability densities are calculated from some unknown test case, i.e., unknown damage class, and the probability of the unknown damage case belonging to a particular damage class is calculated based upon comparison to the known damage cases that were analyzed in pattern training. The general multivariate normal density is written as:

$$p(x) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} exp[-0.5(\mathbf{x}-\boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})] \tag{3.17}$$

where

$$\begin{aligned}
\mathbf{x} &= \text{a d-component column vector} \\
\boldsymbol{\mu} &= \text{a d-component mean vector} \\
&= E[\mathbf{x}] \qquad (3.18) \\
d &= \text{length of the column vector} \\
\boldsymbol{\Sigma} &= \text{is the d-by-d covariance matrix} \\
&= E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^t] \qquad (3.19) \\
E &= \text{the expected value operator}
\end{aligned}$$

If x_i is the i^{th} component of $\mathbf{x}$, μ_i is the i^{th} component of $\boldsymbol{\mu}$, and σ_{ij} is the ij^{th} component of $\boldsymbol{\Sigma}$, then

$$\mu_i = E[x_i] \tag{3.20}$$

$$\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)] \tag{3.21}$$

Swenson (29) showed that the measured FRF data was approximately normally distributed (in a Gaussian manner); therefore, we assumed that to be the case in this research. A simply coded Gaussian classifier was used.

A discriminant function, $g_i(\mathbf{x})$, was first calculated, as follows:

$$g_i(x) = -0.5(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) \tag{3.22}$$

This function represents the distance measure between the test vector, $\mathbf{x}$, and the mean, $\boldsymbol{\mu}_i$, of each damage class. This distance is known as the Mahalanobis distance, not the Euclidean norm. For more details, see Duda and Hart (9). The damage class with the largest discriminant function, i.e., the small negative value, is selected as the damage class present in the structure. The discriminant function was used to identify damage with the speech spectra method. In identifying damage with resonant and anti-resonant frequencies, the Euclidean norm distances were used. The test vector was compared to all known damage classes, and the one producing the smallest Euclidean norm distance was selected as the damaged FTE state. Covariances were not used because, in some cases, resonant or anti-resonant frequencies appeared only once in the training, as discussed previously. In the following chapter, the results from these pattern classifiers will be presented.

IV. Damage Detection Results

4.1 Introduction

The FTE was tested with the pattern classifiers developed in Chapter 3. Both the speech spectra and the resonant frequency methods were tested using measured FTE data different than the data used to train the pattern classifiers themselves. This was done for two reasons. First, it was felt that testing the damage detection algorithm on the same data it was trained upon would not be a rigorous test. Second, and more importantly, it was suspected that the conditions the FTE operate in may change over time. There would be a considerable amount of time between training data collection and the actual testing of the pattern classifiers. It was hoped that lab environment changes would be realized when the damage detection algorithm was tested on new data that may differ from the data gathered for training.

4.2 Resonant and Anti-Resonant Frequencies Damage Identification Results

The resonant and anti-resonant frequencies damage identification algorithm followed the process outlined in Figure 4.1. Using two accelerometer FRFs from the top of the FTE and arranging the resonant and anti-resonant frequencies in 2 Hz intervals from 4-54 Hz, a 1-by-100 vector was created for an unknown FTE condition from the accelerometer data. The first 25 elements of this vector were composed of the x-direction accelerometer resonant frequencies, and elements 26-50 were made up of x-direction accelerometer anti-resonant frequencies. Elements 51-75, and 76-100 were composed of the y-direction resonant and anti-resonant frequencies, respectively. The data that resulted from the training was arranged in an identical fashion for all damage classes. It should be noted that these matrices were sparse. Next, the elements of the non-zero row numbers of the test vector were found and compared, via the Euclidean norm, to the elements in the corresponding rows of the training vectors. The Euclidean norm was calculated from the differences between the test vector and all the training vectors' resonant and anti-resonant frequencies. In addition to computing the Euclidean norms, the damage identification algorithm retained the damage case for each Euclidean norm calculated. The damage case with lowest Euclidean norm computed by the algorithm was reported to be the damage present in the structure.

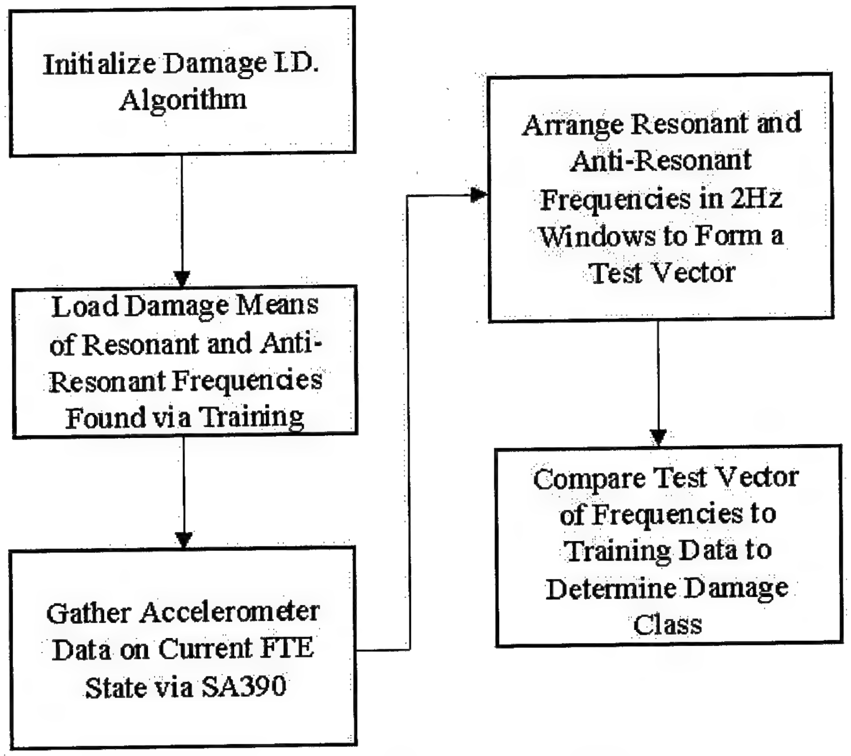

Initialize Damage I.D. Algorithm
Load Damage Means of Resonant and Anti-Resonant Frequencies Found via Training
Gather Accelerometer Data on Current FTE State via SA390
Arrange Resonant and Anti-Resonant Frequencies in 2Hz Windows to Form a Test Vector
Compare Test Vector of Frequencies to Training Data to Determine Damage Class

Figure 4.1 Testing Block Diagram

To reduce computational time, the algorithm was initially tested using only one FRF pair. The results are presented in Table 4.1. Poor results using this approach led us to try using the mean of 10 test FRF pairs. This second method was far superior to those obtained using only one FRF pair; therefore, testing of the one FRF pair method was abandoned after testing only bay 1 of the FTE. The savings in computational time was not significant enough to sacrifice accuracy.

The algorithm was initially trained using only the resonant frequencies to determine damage; then, the algorithm was trained using both resonant and anti-resonant frequencies to determine damage. In addition to the speech spectra method results, tables 4.2, 4.3, 4.4, and 4.5 list results of testing the algorithm using both the resonant and anti-resonant frequencies, where they can be compared to the results obtained using the resonant frequencies alone for damage detection. Thirty FRFs for each damage case were tested using only one FRF pair. With the exception of the speech spectra method, the results shown in Tables 4.3, 4.4, and 4.5 are from testing 10 averages of 10 FRFs (total of 100 FRFs per damage case).

Table 4.1 Algorithm Results Using Only One FRF Pair

Damage State	Success Rate(%)
Baseline, Undamaged	27
Diagonal 1 Removed	100
Diagonal 1, 50% Damaged	40
Diagonal 2 Removed	100
Diagonal 2, 50% Damaged	67
Diagonal 3, Removed	100
Diagonal 3, 50% Damaged	60
Diagonal 4, Removed	100
Diagonal 4, 50% Damaged	67
Diagonals 3, and 4 Removed	33
Diagonals 1, and 4 Removed	100
Diagonals 4, and 2 Removed	100
Diagonals 2, and 3 Removed	93
Diagonals 1, and 2 Removed	93
Diagonals 1, and 3 Removed	100

Table 4.2 Algorithm Results Using Resonant Frequencies Alone and Using Both Resonant & Anti-Resonant Frequencies (10 FRFs), and Speech Spectra Method Results (1 FRF), Bays 1 and 2

Damage State	Resonant Success Rate (%)	Resonant and Anti-Resonant Success Rate (%)	Speech Spectra Method Success Rate (%)
Baseline, Undamaged	50	100	90
Diagonal 1 Removed	100	100	100
Diagonal 1, 50% Damaged	100	100	100
Diagonal 2 Removed	100	100	100
Diagonal 2, 50% Damaged	100	100	100
Diagonal 3, Removed	100	100	100
Diagonal 3, 50% Damaged	100	100	100
Diagonal 4, Removed	100	100	100
Diagonal 4, 50% Damaged	100	100	100
Diagonals 3, and 4 Removed	100	100	100
Diagonals 1, and 4 Removed	100	100	100
Diagonals 4, and 2 Removed	100	100	100
Diagonals 2, and 3 Removed	100	100	100
Diagonals 1, and 2 Removed	100	100	100
Diagonals 1, and 3 Removed	100	100	100
Diagonal 5 Removed	100	100	100
Diagonal 5, 50% Damaged	90	100	100
Diagonal 6 Removed	100	100	100
Diagonal 6, 50% Damaged	100	100	100
Diagonal 7, Removed	100	100	100
Diagonal 7, 50% Damaged	100	100	100
Diagonal 8, Removed	100	100	100
Diagonal 8, 50% Damaged	100	100	100
Diagonals 7, and 8 Removed	100	100	100
Diagonals 5, and 8 Removed	100	100	100
Diagonals 6, and 8 Removed	100	100	100
Diagonals 6, and 7 Removed	100	100	100
Diagonals 5, and 6 Removed	100	100	100
Diagonals 5, and 7 Removed	100	100	100

Table 4.3 Algorithm Results Using Resonant Frequencies Alone and Using Both Resonant & Anti-Resonant Frequencies (10 FRFs), and Speech Spectra Method Results (1 FRF), Bays 3 and 4

Damage State	Resonant Success Rate (%)	Resonant and Anti-Resonant Success Rate (%)	Speech Spectra Method Success Rate (%)
Diagonal 9 Removed	100	100	100
Diagonal 9, 50% Damaged	90	100	100
Diagonal 10 Removed	100	100	100
Diagonal 10, 50% Damaged	100	100	100
Diagonal 11, Removed	100	100	100
Diagonal 11, 50% Damaged	90	100	100
Diagonal 12, Removed	100	100	100
Diagonal 12, 50% Damaged	90	100	100
Diagonals 11, and 12 Removed	100	100	100
Diagonals 9, and 12 Removed	100	100	100
Diagonals 10, and 12 Removed	100	100	100
Diagonals 10, and 11 Removed	100	100	100
Diagonals 9, and 10 Removed	0	100	100
Diagonals 9, and 11 Removed	100	100	100
Diagonal 13 Removed	100	100	100
Diagonal 13, 50% Damaged	60	100	100
Diagonal 14 Removed	100	100	100
Diagonal 14, 50% Damaged	100	100	100
Diagonal 15, Removed	100	100	100
Diagonal 15, 50% Damaged	100	100	100
Diagonal 16, Removed	100	100	100
Diagonal 16, 50% Damaged	80	100	100
Diagonals 15, and 16 Removed	100	100	100
Diagonals 13, and 16 Removed	100	100	100
Diagonals 14, and 16 Removed	100	100	100
Diagonals 14, and 15 Removed	100	100	100
Diagonals 13, and 14 Removed	100	100	100
Diagonals 13, and 15 Removed	100	100	100

Table 4.4 Algorithm Results Using Resonant Frequencies Alone and Using Both Resonant & Anti-Resonant Frequencies (10 FRFs), and Speech Spectra Method Results (1 FRF), Bays 5 and 6

Damage State	Resonant Success Rate (%)	Resonant and Anti-Resonant Success Rate (%)	Speech Spectra Method Success Rate (%)
Diagonal 17 Removed	100	100	100
Diagonal 17, 50% Damaged	10	100	100
Diagonal 18 Removed	100	100	100
Diagonal 18, 50% Damaged	100	100	100
Diagonal 19, Removed	100	100	100
Diagonal 19, 50% Damaged	100	100	100
Diagonal 20, Removed	100	100	100
Diagonal 20, 50% Damaged	80	100	100
Diagonals 19, and 20 Removed	100	100	100
Diagonals 17, and 20 Removed	100	100	100
Diagonals 18, and 20 Removed	100	100	100
Diagonals 18, and 19 Removed	100	100	100
Diagonals 17, and 18 Removed	100	100	100
Diagonals 17, and 19 Removed	100	100	100
Diagonal 21 Removed	100	100	100
Diagonal 21, 50% Damaged	70	100	100
Diagonal 22 Removed	100	100	100
Diagonal 22, 50% Damaged	80	100	100
Diagonal 23, Removed	100	100	100
Diagonal 23, 50% Damaged	100	100	100
Diagonal 24, Removed	100	100	100
Diagonal 24, 50% Damaged	80	100	100
Diagonals 23, and 24 Removed	100	100	100
Diagonals 21, and 24 Removed	100	100	100
Diagonals 22, and 24 Removed	100	100	100
Diagonals 22, and 23 Removed	100	100	100
Diagonals 21, and 22 Removed	100	100	100
Diagonals 21, and 23 Removed	100	100	100

Table 4.5 Algorithm Results Using Resonant Frequencies Alone and Using Both Resonant & Anti-Resonant Frequencies (10 FRFs), and Speech Spectra Method Results (1 FRF), Bays 7 and 8

Damage State	Resonant Success Rate (%)	Resonant and Anti-Resonant Success Rate (%)	Speech Spectra Method Success Rate (%)
Diagonal 25 Removed	90	100	100
Diagonal 25, 50% Damaged	0	100	100
Diagonal 26 Removed	100	100	100
Diagonal 26, 50% Damaged	70	100	100
Diagonal 27, Removed	100	100	100
Diagonal 27, 50% Damaged	0	50	100
Diagonal 28, Removed	100	100	100
Diagonal 28, 50% Damaged	100	100	100
Diagonals 27, and 28 Removed	100	100	100
Diagonals 25, and 28 Removed	100	100	100
Diagonals 26, and 28 Removed	100	100	100
Diagonals 26, and 27 Removed	100	100	100
Diagonals 25, and 26 Removed	100	100	100
Diagonals 25, and 27 Removed	100	100	100
Diagonal 29 Removed	100	100	100
Diagonal 29, 50% Damaged	80	100	100
Diagonal 30 Removed	20	100	100
Diagonal 30, 50% Damaged	100	100	100
Diagonal 31, Removed	100	100	100
Diagonal 31, 50% Damaged	100	100	100
Diagonal 32, Removed	100	100	100
Diagonal 32, 50% Damaged	80	100	100
Diagonals 31, and 32 Removed	100	100	100
Diagonals 29, and 32 Removed	100	100	100
Diagonals 30, and 32 Removed	100	100	100
Diagonals 30, and 31 Removed	100	100	100
Diagonals 29, and 30 Removed	100	100	100
Diagonals 29, and 31 Removed	80	100	100

As Tables 4.2 through 4.5 illustrate, using both anti-resonant and resonant frequencies improved damage detection accuracy. Furthermore, the results indicate that the misdiagnosed damage states were spread uniformly throughout the structure. In other words, the detection algorithm had about the same accuracy regardless of the bay that was damaged. Damage states near the top of the FTE were expected to be less detectable than those near the bottom, because there is less strain energy near the top. The baseline, undamaged configuration is only 50% accurate with the resonant frequency method. This result is undesirable and eliminates the resonant frequency method from consideration for accurate, FTE damage detection.

It was previously mentioned that lab conditions would be examined. It was observed upon initial testing that the baseplate bolts (see Figure 2.2) were loose. Damage detection results improved significantly upon simply tightening the bolts. No other adverse lab conditions were found in testing.

Additional damage detection tests were performed using synthetic training data and tested with measured FRF data. The results are shown in Table 4.6. The FEM generated data was obtained from the work of Jones (17).

Table 4.6 Algorithm Results Using FEM Generated Training Data

Damage State	Success Rate(%)
Baseline, Undamaged	50
Diagonal 1 Removed	100
Diagonal 1, 50% Damaged	0
Diagonal 2 Removed	0
Diagonal 2, 50% Damaged	0
Diagonal 3, Removed	0
Diagonal 3, 50% Damaged	70
Diagonal 4, Removed	0
Diagonal 4, 50% Damaged	0
Diagonals 3, and 4 Removed	0
Diagonals 1, and 4 Removed	100
Diagonals 4, and 2 Removed	100
Diagonals 2, and 3 Removed	40
Diagonals 1, and 2 Removed	100
Diagonals 1, and 3 Removed	100

Results from the other FTE sections were comparable to those seen in Table 4.6. The poor performance indicated by these results can probably be attributed to both inaccuracy of the FEM itself and low discrete data resolution.

4.3 Speech Spectra Method Damage Identification Results

Our pattern classifier testing began with an unsuccessful attempt to duplicate the results obtained by Swenson (29). Swenson used 10 triangular filters per FRF, yielding a feature vector 20 elements long (10 filters X 2 FRFs).

It was decided to increase the resolution associated with the speech spectra method by doubling the number of triangular filters. An accuracy rate better than the one obtained using resonant and anti-resonant frequencies was desired. An accuracy of 90% was achieved in this scenario for the baseline, undamaged configuration, and 100% accuracy was achieved for the 112 remaining damage states, as seen in Tables 4.2, 4.3, 4.4, and 4.5.

The speech spectra method was also trained with FEM generated, synthetic data and tested with measured FRF data. For these results, consult Jones (17).

4.4 Summary of Results

It was shown that the following damage configurations could be correctly identified by the FTE in addition to the undamaged, baseline configuration:

- Complete removal of a diagonal
- Removal of a diagonal and replacement with a partially damaged diagonal
- Complete removal of any two diagonals in one bay

Better damage detection algorithms and better understanding of FTE operation were realized in this research. Both the speech spectra and the resonant/anti-resonant frequencies methods were nearly 100% accurate. However, the latter method proved to be more desirable despite it requiring more computational time, because it always correctly identified the baseline, undamaged configuration. It was also shown that the boundary conditions, i.e., baseplate conditions, impacted damage identification accuracy. Finally, the addition of the anti-resonance frequencies to the algorithm provided the information needed to obtain near 100% accuracy.

In testing the pattern classifier with FEM generated data, results were poor. This can be attributed to poor data point discretization resolution and FEM inaccuracies, as discussed previously. Jones (17) was able to obtain 100% accuracy in the case of complete diagonal removal, but 0% accuracy was achieved in the cases of 50% damage. Results were mixed for the FEM trained pattern classifier used in this work, as seen in Table 4.6.

V. Conclusions and Recommendations

5.1 Conclusions

Several easily programmable algorithms were created that correctly identified damage in a real structure. These algorithms were based on analysis tools from a variety of disciplines, including structural dynamics, speech recognition, and modal analysis techniques. They extracted vectors from FRF magnitude spectra that were shown to vary with damage.

A major drawback of this work was its inherent reliance on a limited number of damage states. Ideally, a damage detection scheme should be able to correctly identify any damage state, the extent of the damage, and the damage location in the structure. To accomplish such detailed damage detection on the FTE, all degrees of freedom would have to be measured, which would require at least 192 accelerometers over the structure. A FTE configuration with this many accelerometers would probably be an unacceptable design for the space environment in which a FTE like structure is intended to operate, since additional payload would have to be sacrificed for sensor weight and associated hardware.

Additionally, most Finite Element Modeling updating tools tend to change all the stiffness matrix values a little, instead of a few significant changes, by updating the assembled matrix itself rather than the physical parameters of the model. Friswell (11) stated that the case of a few significant changes would be typical after damage was introduced into the original, undamaged, baseline, physical structure. Thus, the true nature of the damage would be difficult to detect. This would further hinder a space operator's ability to remotely detect damage.

Near 100% accuracy was obtained with two of the damage detection algorithms generated in this work. Anti-Resonant frequencies were shown to provide excellent results when used with the resonant frequencies. Despite taking steps to eliminate leakage and aliasing in the experimental data-acquisition set-up, these may have caused errors in addition to the sources stated previously. An additional source of error that should be considered in any future FTE research is the FTE baseplate mount. A more secure mounting design should be implemented.

Results from training with FEM generated data are not encouraging. Only complete diagonal removal was successfully diagnosed by Jones (17) and no consistent results were obtained in this work with FEM trained pattern classifiers.

5.2 Recommendations

Future research on the FTE or any other structure should consider using anti-resonant frequencies in their damage detection schemes. They were shown to be effective damage indicators in this work, because they vary by measurement location (unlike resonant frequencies, that are global to the structure). More sensors could be used to determine additional anti-resonant frequencies in a structure, which in turn would provide insight into more damage states.

Appendix A. Matlab Code

A.1 Speech Spectra Damage Detection Training Matlab Files

```
%************************************************************************
% Matlab File: gtraintf.m
%************************************************************************
% Created by:  Capt Arb
% Date: 12 Aug 97
% Modified by: Capt Gaeta
% Date: 1 Nov 99
% This file created the training features from the raw FRF data.
% This file used during training and not during the damage ID process.
% The raw data is passed thru triangular filters producing feature sets.
% The output of these filters are known as Mel-Frequency Spectral Coefficients,
% or MFSC's.  These MFSC's are then projected onto a set of orthogonal cosines
% to produce Mel-Frequency Cepstral Coefficients, MFCC's.

function [train_ftrs]=gtraintf(memberlist);

train_ftrs=[];


for k=1:memberlist;
    nextmember=k-1;
    nextmember
    filename=['tdmg' int2str(nextmember)];
    eval(['load ' filename]);
    eval(['xdata=y36' int2str(nextmember) ';']);
    eval(['ydata=x33' int2str(nextmember) ';']);
    xdata=[(k-1)*ones(size(xdata,1),1) 10*log10(xdata)];
    ydata=[(k-1)*ones(size(ydata,1),1) 10*log10(ydata)];
    xftrs=tfiltdct(xdata);
    yftrs=tfiltdct(ydata);
    ftrs=[xftrs yftrs(:,2:size(yftrs,2))];
    train_ftrs=[train_ftrs;ftrs];

end;

train_ftrs=[train_ftrs(:,1:21) train_ftrs(:,23:42)];
```

```
%************************************************************************
% Matlab File: tfiltdct.m
%************************************************************************

function features=tfiltdct(labelled_data);
%
% Author:   Capt Al Arb
% Date:     8 Aug 97

% Modified: Capt Doug Gaeta
% Date:     1 Sept 99
%
% This function performs passes the input data (assumed to be a spectrum)
% through a bank of triangular filters.  The triangular filters are spaced
% linearly from 0-50 Hz with 50% overlap.  From there they are spaced
% logaritmically.  The gain of each is 1.
%
% The output of the filter is the sum of the weighted spectrum and is
% called the Mel-Frequency Spectral Coefficient (MFSC).  The vector of
% MFSC's is then projected onto a set of orthogonal cosines through the
% DCT to produce Mel-Frequency Cepstral Coefficients.
%
% The data is assumed to be labelled with the first column being
% class label, and the rest of each row the specturm.
%
% The code can be easily modified to allow for some parameters to be passed in
% instead of hard coded. (e.g. melSpacing, linear cutoff frequency, fs,
% etc.)

size(labelled_data,2);

data=labelled_data(:,2:size(labelled_data,2)-1);

size(data);

fs=200;
melSpacing=50;
mels=melSpacing;
%
% convert digital frequency fs/2 to mel scale
%
maxmels=2595*log10(1+fs/50/2); spectsize=400; k=1;
%
% find center frequencies on mel scale for filters
%
while mels(k)<maxmels,
   mels=[mels mels(k)+melSpacing];
   k=k+1;
end;

melsize=size(mels);
%
% Convert back to analog frequency
% Finding lower, center, and upper frequencies for filters
%
fc=((10.^(mels./2595))-1).*50;
lower=[0 fc(1:k-2)];
center=fc(1:k-1);
upper=fc(2:k);

numfilters=k-1;
% There may only be k-2 complete filters

filtweights=zeros(numfilters,spectsize);
%
% Define triangular filters:
%
%              |--------------------------------------------|
%              |  /\                                         |
%            1 |_/  \________________________________________|
%              |    /\                                       |
%            2 |___/  \______________________________________|
%                                   :
%%                    :
%              |                                        /\  |
%   numfilters |_______________________________________/  \_|
%              |--------------------------------------------|

for l=1:numfilters
   for k=1:spectsize,
      if ((k >= (lower(l)/(fs/2)*spectsize))&(k <= (center(l)/(fs/2)*spectsize)))
     filtweights(l,k)=((k/spectsize)*(fs/2)-lower(l))/(center(l)-lower(l));
      elseif ((k > center(l)/(fs/2)*spectsize)&(k <= upper(l)/(fs/2)*spectsize))
     filtweights(l,k)=(upper(l)-(k/spectsize)*(fs/2))/(upper(l)-center(l));
      end;
   end;
end;

%
% Examine last filter.  If it's not complete enough (i.e., only part of it
% may be below fs/2), then we discard the last filter.
%
```

```
if filtweights(numfilters,spectsize)>0.1
    numfilters=numfilters-1;
    filtweights=filtweights(1:numfilters,:);
end;

%
% a is a normalizing factor to ensure the gain is 1 for each filter
%
a=sum(filtweights')';

%
% initialize new feature matrix
%
features=zeros(size(data,1),numfilters);


%
% Calculate MFSC's
%
     mfb=data*(filtweights.*[a.^(-1)*ones(1,size(filtweights,2))])';


%
% Do DCT
%
     features=dctarb(mfb);

     %features

features=[labelled_data(:,1) features];

%**********************************************************************
% Matlab File: dctarb.m
%**********************************************************************


function b=dctarb(a,n)
% Discrete cosine transform.
%
% Y = DCT(X) returns the discrete cosine transform of X.
% The vector Y is the same size as X and contains the
% discrete cosine transform coefficients.
%
% Y = DCT(X,N) pads or truncates the vector X to length N
% before transforming.
%
```

```
% If X is a matrix, the DCT operation is applied to each
% column.  This transform can be inverted using IDCT.
%
% See also: FFT,IFFT, and IDCT.

% Author(s): C. Thompson, 2-12-93
% Copyright (c) 1984-94 by The MathWorks, Inc.
% Revision: 1.8 $  $Date: 1994/01/25 17:58:57 $

% Revised by Al Arb to conform to DCT operation in HTK
%
% References:
% Jae S. Lim, "Two-dimensional Signal and Image Processing",
% pp. 148-162.  Implements an even-symmetrical DCT.
% Jain, "Fundamentals of Digital Image Processing", pp. 150-153.
% Wallace, "The JPEG Still Picture Compression Standard",
% Communications of the ACM, April 1991.

% The input to this file are the MFSC's and the output is a vector 40 units long
% known as MFCC's.

error(nargchk(1,2,nargin));


if min(size(a))==1
    if size(a,2)>1
        do_trans = 1;
    else
        do_trans = 0;
    end
    a = a(:);
else
    do_trans = 0;
end if nargin==1,
  n = size(a,1);
end m = size(a,2);

% Pad or truncate a if necessary
if size(a,1)<n,
  aa = zeros(n,m);
  aa(1:size(a,1),:) = a;
else
  aa = a(1:n,:);
end
```

```
%
% HTK defines DCT as:
%
% c_i=sqrt(2/N) sum(j=1:N) {m_j*cos(((pi*i)/N) * (j-0.5))}
%
% Thus, c=sqrt(2/N).*(aa*COSMTRX)
%
% where COSTMTRX=|-------------------------------------------------|
%                | cos(((pi*1/N)*(1-0.5)) ... cos(((pi*N/N)*(1-0.5))|
%                | cos(((pi*1/N)*(2-0.5)) ... cos(((pi*N/N)*(2-0.5))|
%                |             :              :        |
%                | cos(((pi*1/N)*(N-0.5)) ... cos(((pi*N/N)*(N-0.5))|
%                |-------------------------------------------------|
%
% and N= the number of coefficients/elements in X
%

N=size(aa,2);
COSMTRX=cos((pi/N).*[ones(N,1)*[1:N]].*[[1:N]'*ones(1,N)-0.5]);
b=sqrt(2/N).*(aa*COSMTRX);

% original stuff:

% if rem(n,2)==1, % odd case
% Form intermediate even-symmetric matrix.
% y = zeros(2*n,m);
% y(1:n,:) = aa;
% y(n+1:n+n,:) = flipud(aa);

% Perform FFT
% yy = fft(y);

% Compute DCT coefficients
% ww = exp(-sqrt(-1)*(0:n-1)*pi/(2*n)).';
% b = ww(:,ones(1,m)).*yy(1:n,:);

% else % even case, courtesy of Steven L. Eddins

% Re-order the elements of the columns of x
% y = [ aa(1:2:n,:); aa(n:-2:2,:) ];

% Compute weights to multiply DFT coefficients
% ww = 2*exp(-sqrt(-1)*(0:n-1)'*pi/(2*n));
% W = ww(:,ones(1,m));

% Compute DCT using equation (5.92) in Jain
% b = W .* fft(y);
% end

if isreal(a), b = real(b); end if do_trans, b = b.'; end

%*************************************************************************
% Matlab File: gtrain.m
%*************************************************************************

% Baseline Gaussian Classifier.
% Created by:  CPT Pellissier.
% Modified by: Capt Arb and Capt Gaeta
% Date: 1 Nov 99
% This script uses a Gaussian Classifier with a common
% (but full) covariance matrices for each class.
% This file is used for training in the speech spectra method.
% The input to this file is the output of gtraintf.m, i.e., the Mel-Frequency
% Cepstral Coefficients. The outputs are the damage class means and the covariances.

function [trainMean,sigma]=gtrain(tdata);

% initialization

  data=tdata; clear tdata;
  [dataRow,dataCol] = size(data);
  numTrainClasses = max(data(:,1))+1;  % number of training classes
  features = data(:,2:dataCol);
  trainClasses = 0:numTrainClasses-1;

% Compute the class means and covariance from the training set

  trainMean = zeros(numTrainClasses,dataCol-1);
  sigma = zeros(dataCol-1);
  for i = 1:numTrainClasses,
    lineNumbers = find(data(:,1) == trainClasses(i));
    numberLines = length(lineNumbers);
    if numberLines > 1
         trainMean(i,:) = mean(features(lineNumbers,:));
    else
         trainMean(i,:)=features(lineNumbers,:);
    end;
  end;
  sigma=diag(diag(cov(features)));
  save c:\gaeta\damageid\sigtrm5.mat;
```

A.2 Speech Spectra Damage Detection Testing Matlab Files

```
%************************************************************************
% Matlab File: readddmg.m
%************************************************************************

% This file executes the entire damage ID process.  It starts by
% loading the output of the training, sigmatrm.mat which contain
% the variables sigma and trainmean.  Then it
% collects data on the current state of the truss via SA390 and mfile
% readtest. After that the data is reduced from [1X400] to a [1X40]
% via triangular filtering.  This is done with gentest file that
% contains the filtering process.  From there, a statiscal process
% is executed that determines the damage class from the feature set.

clear all

load C:\gaeta\damageid\sigtrm*.mat

% this .mat file contains the variables
% sigma and trainmean, which are the covariances and means for all 113 damage states.

[c0,d0]=readtest;
% Gathers FRF data from current FTE state.
test=gentest(c0,d0);
%
gtest(trainMean,sigma,test);


%************************************************************************
% Matlab File: readtest.m
%************************************************************************
% Created by Capt Swenson
% Modified by Capt Gaeta
% Date: 1 Sep 99
% This file initializes the SA 390 and converts the two trace files
% generated from accelerometer at nodes 33(X) and node (36) into MATLAB
% format-traceA.trc and traceB.trc.

function [c0,d0]=readtest(); c = ddeinit('sd390','COMMAND');

cr = ddeexec(c,'CTRL AVGR START'); c = ddeinit('sd390','STATUS');
cr = ddereq(c,'INFO'); while fix(cr/128) == 0
  cr = ddereq(c,'INFO');
  pause(1)
end c = ddeinit('sd390','COMMAND');
```

```
cr = ddeexec(c,'SAVEtrcAc:\gaeta\damageid\traceA.trc'); pause(1)
cr = ddeexec(c,'SAVE trcB c:\gaeta\damageid\traceB.trc'); pause(1)
fid=fopen('c:\gaeta\damageid\traceA.trc');
status=fseek(fid,676,0); [A,count]=fread(fid,'float');
size(count); status=fclose('all');
fid=fopen('c:\gaeta\damageid\traceB.trc');
status=fseek(fid,676,0); [B,count]=fread(fid,'float');
status=fclose('all'); c0=B'; d0=A';

%*********************************************************************
% Matlab File: gentest.m
%*********************************************************************
%
% Created by Capt Gaeta
% Date: 1 Sep 99
% This file takes the raw, unknown, FRF magnitude data, and produces the
% MFCC's as a result of both the filtering and discrete cosine transform processes.

function [test_ftrs]=gentest(xdata,ydata);

test_ftrs=[];

xdata=[0 10*log10(xdata)]; ydata=[0 10*log10(ydata)];
xdata=[xdata; xdata]; ydata=[ydata ; ydata];
xftrs=tfiltdct(xdata); yftrs=tfiltdct(ydata);
%
test_ftrs=[xftrs yftrs(:,2:size(yftrs,2))]; size(yftrs,2);
test_ftrs=[test_ftrs(1,1:21) test_ftrs(1,23:42)];

%*********************************************************************
% Matlab File: gtest.m
%*********************************************************************
% Baseline Gaussian Classifier.
% Created by: CPT Pellissier.
% Modified by: Capt Gaeta
% Date: 1 Sep 99
% This script uses a Gaussian Classifier with a common
% (but full) covariance matrices for each class.
%
function result=gtest(trainMean,sigma,test_ftrs);

testData = test_ftrs(:,2:length(test_ftrs));

% Classify
% trainMean is a [number of damage cases X 40] size matrix, i.e., currently,
% 113 unique damage cases by 40 feature vectors.  The upper index on the below for loop
% shall always be the total number of damage cases.

for i=1:size(trainMean,1),%returns number of rows, which is equal to number of damage cases-33
    g(i,1)=-(0.5)*1/(mean(diag(sigma)))*(testData-trainMean(i,:))*(testData-trainMean(i,:))';
end;
    [maxg,classIndex] = max(g);
    member = classIndex-1;
    if member<=56
      load c:\gaeta\damageid\class1.mat;
      damage=class1(member*14+1:(14*(member+1)));
      damage
    end;
    if member>56
      load c:\gaeta\damageid\class2.mat;
      damage=class2((member-57)*14+1:(14*((member-57)+1)));
      damage
    end;
```

A.3 Resonant & Anti-Resonant Frequencies Damage Detection Training Matlab Files

```
%***********************************************************************
% Matlab File: gaeta1.m
%***********************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file loads the raw FRF data, takes the mean of 10 FRF's, and sends it
% to a second m-file to extract the resonant and anti-resonant frequencies for
% both accelerometers.

clear;

for i=1:14;
   eval(['load f:\readtrng\section1\tdmg'int2str(i)]);
   eval(['x=x33' int2str(i) ';'])
   eval(['y=y36' int2str(i) ';'])
   for j=0:9;
       xav(j+1,:)=mean(x(j*10+1:(j+1)*10,:));
       yav(j+1,:)=mean(y(j*10+1:(j+1)*10,:));
    end;
   size(xav);
   clear x;
   clear y;
   x=xav;
   y=yav;
   [wp,wz,peakvect,valleyvect,polemag,polefreq,zeromag,zerofreq]=arb2(x,i,1);
   [wp,wz,peakvect,valleyvect,polemag,polefreq,zeromag,zerofreq]=arb2(y,i,2);
   i
end;

%***********************************************************************
% Matlab File: arb2.m
%***********************************************************************
% Created by: Al Arb
% Modified by: Capt Gaeta
% Date: 1 Dec 99
% This file extracts the resonant & anti-resonant frequencies for the x and y
% acclerometers.
% The optimal value of jitter/window length is 5, this value picks up local peaks & valleys.
% N (number of data points) of 241 corresponds to cutoff FRF frequency of 60Hz.
% input xory is used to distinguish between accel. measurements in the x or y direction,
% xory of 1 corresponds to x, xory of 2 corresponds to y.
```

```
function [wp,wz,peakvect,valleyvect,polemag,polefreq,zeromag,zerofreq]...
         =get_formants(magspect,class,xory);
jitter=5;
N=241;
frequency=linspace(0,60,241)';

wp=frequency; wz=frequency;
len=length(wp); magspect=magspect';
magspect=10*log10(magspect);


% input data, magspect, is composed of data up to 100Hz.
% wp(0-60Hz) will be composed of the frequencies corresponding to the peaks.
% wz(0-60Hz) will be composed of the frequencies corresponding to the valleys.
% beyond 60Hz a typical FRF taken from the FTE has lots of noise.
%
% assume magspect=10*log10(abs(spect)), i.e., power domain
%

numframes=length(magspect(1,:));

peakvect=magspect;
valleyvect=magspect;
count=0; %  initialize count
ticker=0;

for F=1:numframes
   for j=jitter+1:N-jitter
   ticker=ticker+1;
   [peak, peakbin]=max((magspect(j-jitter:j+jitter,F)));

   % defined variable peakbin, retains the index where the max was found

   peak=peak(1);
   peakbin=peakbin(1);
   count=count+1;
      if peakbin==1,
            peakvect(j-jitter+1:j+jitter,F)=-1000*ones(2*jitter,1);
            wp(j-jitter+1:j+jitter,F)=-1000*ones(2*jitter,1);

      % above sets peakvect equal to -1000 if the peak occured in the first
      % element analyzed

      elseif peakbin == 2*jitter+1,
            peakvect(j-jitter:j+jitter-1,F)=-1000*ones(2*jitter,1);
            wp(j-jitter:j+jitter-1,F)=-1000*ones(2*jitter,1);

      % above sets peakvect equal to -1000 if peak occured in the last
      % element analyzed in the max statement above.

      else
            peakvect(j-jitter:j-jitter+peakbin-2,F)=...
            -1000*ones(peakbin-1,1);
            wp(j-jitter:j-jitter+peakbin-2,F)=...
            -1000*ones(peakbin-1,1);

        peakvect(j-jitter+peakbin:j+jitter,F)=...
            -1000*ones(2*jitter+1-peakbin,1);
            wp(j-jitter+peakbin:j+jitter,F)=...
            -1000*ones(2*jitter+1-peakbin,1);

            %above sets peakvect equal to -1000, if the peak occurs anywhere
            %other than the first or last point in the 'window'

     end;

end;

end;
% for loop F
count;

% The following is a repitition of the above to find the zeroes

for G=1:numframes for k=jitter+1:N-jitter;

[valley, valleybin]=min((magspect(k-jitter:k+jitter,G)));
valley=valley(1); valleybin=valleybin(1);

      if valleybin==1,
            valleyvect(k-jitter+1:k+jitter,G)=-1000*ones(2*jitter,1);
            wz(k-jitter+1:k+jitter,G)=-1000*ones(2*jitter,1);

      elseif valleybin == 2*jitter+1,
            valleyvect(k-jitter:k+jitter-1,G)=-1000*ones(2*jitter,1);
            wz(k-jitter:k+jitter-1,G)=-1000*ones(2*jitter,1);

      else
            valleyvect(k-jitter:k-jitter+valleybin-2,G)=...
            -1000*ones(valleybin-1,1);
            wz(k-jitter:k-jitter+valleybin-2,G)=...
            -1000*ones(valleybin-1,1);

        valleyvect(k-jitter+valleybin:k+jitter,G)=...
            -1000*ones(2*jitter+1-valleybin,1);
            wz(k-jitter+valleybin:k+jitter,G)=...
            -1000*ones(2*jitter+1-valleybin,1);

    end;

end;

end;

% for loop G
%
% the following extracts the pole locations and magnitudes from the data:
%
for h=1:numframes; tick1=0;
% initialize counter variable tick1.
  for j=1:N;
    if peakvect(j,h)>-1000
       tick1=tick1+1;
       polemag(tick1,h)=peakvect(j,h);
       polefreq(tick1,h)=frequency(j,1);
    end;  % ends if statement
  end;  % ends for statement with index j
end;  % ends for statement with index h
%
% the following extracts the zero locations and magnitudes from the data.
%
for m=1:numframes; tick2=0;
% initialize counter variable tick2.
  for p=1:N;
    if valleyvect(p,m)>-1000
      tick2=tick2+1;
      zeromag(tick2,m)=valleyvect(p,m);
      zerofreq(tick2,m)=frequency(p,1);
    end;  % ends if statement
  end;  % ends for statement with index p
end;  % ends for statement with index m
if xory==1
  eval(['pmagx' int2str(class) '=polemag;']);
  eval(['pfreqx' int2str(class) '=polefreq;']);
  eval(['zmagx' int2str(class) '=zeromag;']);
  eval(['zfreqx' int2str(class) '=zerofreq;']);
  eval(['save f:\newclass\section1\xdmg' int2str(class) ' pfreqx' int2str(class) ' ...
  pmagx' int2str(class) 'zmagx' int2str(class) ' zfreqx' int2str(class);]);
end;

if xory==2
  eval(['pmagy' int2str(class) '=polemag;']);
  eval(['pfreqy' int2str(class) '=polefreq;']);
  eval(['zmagy' int2str(class) '=zeromag;']);
  eval(['zfreqy' int2str(class) '=zerofreq;']);
  eval(['save f:\newclass\section1\ydmg' int2str(class) ' pfreqy' int2str(class) ' ...
  pmagy' int2str(class} ' zmagy' int2str(class) ' zfreqy' int2str(class);]);
end;

%**********************************************************************
% Matlab File: xcheckingfrequencyranges.m
%**********************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file is used to determine if more than one resonant frequencies exists in
% a particular frequency range, i.e., is there more than one resonant frequency from
% 4-6.50 Hz?  If yes, the input, window, will be changed until there is either one or zero
% resonant frequency in a range.  Nearly identical code was written for the y accelerometer
% resonant & anti resonant frequencies, and the x accelerometer anti resonant frequencies.
% The output from arb2.m is the input to this routine.

function []=xcheckingfrequencyranges(window,zeroor1)
for i=zeroor1:14;
   eval(['load d:\newclass\section8\xdmg'int2str(i)]);
   eval(['px=pfreqx' int2str(i) ';'])
   [A,B]=size(px);
   for j=1:10;
   %looping thru the 10 columns that are averaged FRF's

      for k=1:26
      %looping thru frequecy 4-54 Hertz
        countpx=0;
        for l=1:A;
      %looping thru each row of each column

            if px(l,j)>=window*k
              if px(l,j)<window*(k+1)
               countpx=countpx+1;
              end;
              if countpx>1
                 freqinquestion=window*k;
                 row=l;
                 column=j;
```

```
                file=i;
                file,freqinquestion,column
             end;
           end;%ending 1st if
        end;%ending last for loop
     end;%ending third for loop
  end;%ending second for loop
end;%ending first for loop.
%***********************************************************************
% Matlab File: arrangepx.m
%***********************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file arranges the extracted x-accelerometer resonant frequencies into
% ranges determined by the input variable window.  This arrangement will allow
% for consistent comparison between a test vector for some unknown damage state,
% and the training data.  A nearly identical file was written for the other resonant
% and anti-resonant frequencies.  The input data to this file is the output of arb2.m.

function []=arrangepx(window,zeroor1)
for i=zeroor1:14;
  eval(['load f:\newclass\section8\xdmg'int2str(i)]);
  %loading files that contain poles and zeroes-x acclerometer only
  eval(['px=pfreqx' int2str(i) ';'])
  [A,B]=size(px);
  for j=1:10;
  %looping thru the 10 columns that are averaged FRF's

     for l=1:A;
     %looping thru each row
         for k=2:26
         %looping thru frequencies 4-54 Hz
          if px(l,j)>0
            if px(l,j)>=window*k
              if px(l,j)<window*(k+1)
                pxarr(k-1,j)=px(l,j);
              end;
            else
              pxarr(k-1,j)=0;
            end;%ending 2nd if
          end;%ending 1st if
         end;%ending last for loop
     end;%ending third for loop
  end;%ending second for loop
  eval(['pxar' int2str(i) '=pxarr;']);
  eval(['save f:\newclass\section8\pxar' int2str(i) ' pxar' int2str(i)]);
```

```
end;%ending first for loop

%************************************************************************
% Matlab File: pxmeanco.m
%************************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file computes means and covariances of the x-accelerometer resonant
% frequencies.  The output of arrangepx.m is the input.  A nearly identical file
% was written for the other resonant and anti-resonant frequencies.

function []=pxmeanco(zeroor1)
%
for i=zeroor1:14;
   frqcov=zeros([25 1]);
   frqmean=zeros([25 1]);
   %looping thru varying FTE config data files.
   eval(['load f:\newclass\section1\pxar'int2str(i)]);
   eval(['px=pxar' int2str(i) ';']);
   [rows,columns]=size(px);
   [test1,test2,test3]=find(px);
   for j=1:rows
      %looping thru each row where each row is a frequency band of 2 Hz
      %
      for k=1:10
         %looping thru each column where each column is a mean of 10 FRF's
         %
         if px(j,k)>0;
            [l,m,frequencies]=find(px(j,1:10));
            frqmean(j,1)=mean(frequencies);
            frqcov(j,1)=cov(frequencies);
         end;
      %
      end;%k-loop
   end %j-loop
   eval(['xpfreqmean' int2str(i) '=frqmean;']);
   eval(['xpfreqcov' int2str(i) '=frqcov;']);
   eval(['save f:\newclass\section1\pxmnco' int2str(i) ' xpfreqmean' int2str(i) '...
   xpfreqcov' int2str(i)]);
end;%i-loop
```

A.4 Resonant & Anti-Resonant Frequencies Damage Detection Testing Matlab Files

```
%*************************************************************************
% Matlab File: identify.m
%*************************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file executes the entire damage ID process for the resonant and anti-resonant
% frequencies method. As was done previously, it starts by loading the output of the training
% then it gathers data on the current state of the FTE.  It gathers 10 FRF's and takes
% their mean.  A statistical analysis is made to determine which member is damaged after
% the test vector is arranged in a manner consistent with the training data output.

function []=identify()
   for i=1:10;
     [x0(i,:),y0(i,:)]=getest;
     size(y0);
     % gathering 10 FRF's to take mean
   end;
    [xpolefreq,xzerofreq]=mkvectx(mean(x0(1:10,:)),1);
    [ypolefreq,yzerofreq]=mkvecty(mean(y0(1:10,:)),2);
    [xpolearr,xzeroarr,ypolearr,yzeroarr,test_vector]...
       =arrgfreq(xpolefreq,xzerofreq,ypolefreq,yzerofreq,2);
    load c:\gaeta\newclass\meancov4
    mnarr(:,107)-test_vector;
    [answer]=newtest(test_vector)
    %
    [maxg,classIndex] = max(answer);
    %classIndex returns the column where the max(answer) occurs
    member = classIndex-1;
    %since damage case zero is actually the first row-to get the correct
    %damage case subtract by 1
    if member<=56
      load c:\gaeta\damageid\class1.mat;
      damage=class1(member*14+1:(14*(member+1)));
      damage
    end
    if member>56
      load c:\gaeta\damageid\class2.mat;
      damage=class2((member-57)*14+1:(14*((member-57)+1)));
      damage
    end;

%*************************************************************************
```

```
% Matlab File: getest.m
%***********************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file computes means and covariances of the x-accelerometer resonant
% frequencies.  The output of arrangepx.m is the input.  A nearly identical file
% was written for the other resonant and anti-resonant frequencies.
% This file initializes the SA 390 and converts the two trace files
% generated from accelerometer at nodes 33(X) and node (36) into MATLAB
% format-traceA.trc and traceB.trc.
```

```
% This file initializes the SA 390 and converts the two trace files
% generated from accelerometer at nodes 33(X) and node (36) into MATLAB
% format-traceA.trc and traceB.trc.
```

```
function [x0,y0]=getest();
c = ddeinit('sd390','COMMAND');
cr =ddeexec(c,'CTRL AVGR START');
c = ddeinit('sd390','STATUS');
cr = ddereq(c,'INFO');

while fix(cr/128) == 0
  cr = ddereq(c,'INFO');
  pause(1)
end;

c = ddeinit('sd390','COMMAND');

cr = ddeexec(c,'SAVE trcA c:\gaeta\damageid\traceA.trc'); pause(1)
cr = ddeexec(c,'SAVE trcB c:\gaeta\damageid\traceB.trc'); pause(1)

fid=fopen('c:\gaeta\damageid\traceA.trc');
status=fseek(fid,676,0);
[A,count]=fread(fid,'float'); size(A);
size(count); status=fclose('all');

fid=fopen('c:\gaeta\damageid\traceB.trc');
status=fseek(fid,676,0);
[B,count]=fread(fid,'float');
status=fclose('all'); y0=B'; x0=A';

%***********************************************************************
% Matlab File: mkvectx.m
%***********************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file extracts the resonant & anti-resonant frequencies for the x
% acclerometer for an unknown test vector.  A nearly identical program was
% written for the y-accelerometer.

function [polefreq,zerofreq]=mkvectx(magspect,xory);

jitter=5;
N=241;
frequency=linspace(0,60,241)'; size(frequency);
wp=frequency;
wz=frequency;
len=length(wp);
magspect=magspect';
magspect=10*log10(magspect); size(magspect);
numframes=length(magspect(1,:));
peakvect=magspect;
valleyvect=magspect;
count=0; %  initialize count
ticker=0;

for F=1:numframes
   for j=jitter+1:N-jitter
   ticker=ticker+1;
   [peak, peakbin]=max((magspect(j-jitter:j+jitter,F)));

   % defined variable peakbin, retains the index where the max was found
   % in above statement.

   peak=peak(1);
   peakbin=peakbin(1);
   count=count+1;
      if peakbin==1,
            peakvect(j-jitter+1:j+jitter,F)=-1000*ones(2*jitter,1);
            wp(j-jitter+1:j+jitter,F)=-1000*ones(2*jitter,1);

      % above sets peakvect equal to -1000 if the peak occured in the first
      % element analyzed in the max statement above.

      elseif peakbin == 2*jitter+1,
            peakvect(j-jitter:j+jitter-1,F)=-1000*ones(2*jitter,1);
            wp(j-jitter:j+jitter-1,F)=-1000*ones(2*jitter,1);

      % above sets peakvect equal to -1000 if peak occured in the last
      % element analyzed in the max statement above.

      else
```

```
            peakvect(j-jitter:j-jitter+peakbin-2,F)=...
            -1000*ones(peakbin-1,1);
            wp(j-jitter:j-jitter+peakbin-2,F)=...
            -1000*ones(peakbin-1,1);

        peakvect(j-jitter+peakbin:j+jitter,F)=...
            -1000*ones(2*jitter+1-peakbin,1);
            wp(j-jitter+peakbin:j+jitter,F)=...
            -1000*ones(2*jitter+1-peakbin,1);

            %above sets peakvect equal to -1000, if the peak occurs anywhere
            %other than the first or last point in the 'window'

    end;

end;

end; % for F
count;

% The following is a repitition of the above to find the zeroes

for G=1:numframes for k=jitter+1:N-jitter;

[valley, valleybin]=min((magspect(k-jitter:k+jitter,G)));
valley=valley(1); valleybin=valleybin(1);

      if valleybin==1,
            valleyvect(k-jitter+1:k+jitter,G)=-1000*ones(2*jitter,1);
            wz(k-jitter+1:k+jitter,G)=-1000*ones(2*jitter,1);

      elseif valleybin == 2*jitter+1,
            valleyvect(k-jitter:k+jitter-1,G)=-1000*ones(2*jitter,1);
            wz(k-jitter:k+jitter-1,G)=-1000*ones(2*jitter,1);

      else
            valleyvect(k-jitter:k-jitter+valleybin-2,G)=...
            -1000*ones(valleybin-1,1);
            wz(k-jitter:k-jitter+valleybin-2,G)=...
            -1000*ones(valleybin-1,1);

        valleyvect(k-jitter+valleybin:k+jitter,G)=...
            -1000*ones(2*jitter+1-valleybin,1);
            wz(k-jitter+valleybin:k+jitter,G)=...
            -1000*ones(2*jitter+1-valleybin,1);
    end;

end;

end; % for G
%
% the following extracts the pole locations and magnitudes from the data:
%
for h=1:numframes; tick1=0;
% initialize counter variable tick1.
  for j=1:N;
    if peakvect(j,h)>-1000
       tick1=tick1+1;
       polemag(tick1,h)=peakvect(j,h);
       polefreq(tick1,h)=frequency(j,1);
    end;  % ends if statement
  end;  % ends for statement with index j
end;  % ends for statement with index h
%
% the following extracts the zero locations and magnitudes from the data.
%
for m=1:numframes; tick2=0;
% initialize counter variable tick2.
  for p=1:N;
    if valleyvect(p,m)>-1000
      tick2=tick2+1;
      zeromag(tick2,m)=valleyvect(p,m);
      zerofreq(tick2,m)=frequency(p,1);
    end;  % ends if statement
  end;  % ends for statement with index p
end;  % ends for statement with index m
if xory==1
  pfreqx=polefreq;
  zfreqx=zerofreq;
end

if xory==2
  pfreqy=polefreq;
  zfreqy=zerofreq;
end

%***********************************************************************
% Matlab File: newtest.m
%***********************************************************************
% Created by: Capt Gaeta
% Date: 1 Dec 99
% This file statistically analyzes the unknown, test vector to the training
% data to determine the damage state the FTE is currently in.

function [answer]=newtest(test_vector);
  load c:\gaeta\newclass\meancov4.mat
  %contains variables mnarr and covarr both [100 by 113]
  [A,numclasses]=size(mnarr);
  %
  %numclasses is variable that is equal to the number of damage classes
  %
  [rows,columns,v] = find(test_vector);
  b=length(rows);
  %
  for i=1:numclasses;
  %
      for j=1:b;
       if rows(j)==1;
          gathermeans(j,i)=mnarr(1,i);
          gathercovs(j,i)=covarr(1,i);
          elseif rows(j)==100;
          gathermeans(j,i)=mnarr(100,i);
          gathercovs(j,1)=covarr(100,i);
       else
          gathermeans(j,i)=mnarr(rows(j),i);
          gathercovs(j,i)=covarr(rows(j),i);
       end;%end if statement
          vector(j,1)=v(j,1);
     end;  %end for loop with index j
     %
     [C,D,E] = find(gathercovs(:,i));
     gathermeans;
     F=length(C);
       answer(1,i)=-0.5*...
       (vector(:,1)-gathermeans(:,i))'*(vector(:,1)-gathermeans(:,i));
  end;  %end for loop with index i
```

Bibliography

1. Afolabi, D. "An Anti-Resonance Technique For Detecting Structural Damage," Proceedings of the Fifth International Modal Analysis Conference, 491–495 (1987).

2. Agneni, A. and others. "Damage Detection on Aeronautical Structures by a Mixed Approach in the Frequency Domain," Proceedings of the 14th International Modal Analysis Conference, 1415–1422 (1996).

3. Agnes, G. "MECH719: Vibration, Damping, and Control Class Notes." Air Force Institute of Technology, Summer 1999.

4. Atalla, M. *Model Updating Using Neural Networks*. PhD dissertation, Virginia Polytechnic Institute and State University, Blacksburg VA, 1996.

5. Cobb, R. *Structural Damage Identification From Limited Measurement Data*. PhD dissertation, Air Force Institute of Technology, Wright-Patterson AFB OH, 1996.

6. Colombi, J.M. *Generalized Hidden Filter Markov Models Applied to Speaker Recognition*. PhD dissertation, Air Force Institute of Technology, Wright-Patterson AFB OH, 1996.

7. Cox, A.M. *A Statistical Analysis of Space Structure Mode Localization*. MS thesis, Air Force Institute of Technology, Wright-Patterson AFB OH, 1999.

8. Davis, S.B. and P. Mermelstein. "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," IEEE Transactions on Acoustics, Speech, and Signal Processing, 28(4):357–366 (August 1980).

9. Duda, R.O. and P.E. Hart. *Pattern Classification and Scene Analysis*. New York: John Wiley and Sons, 1973.

10. Elkordy, M. and others. "Neural Network Trained by Analytically Simulated Damage States," ASCE Journal of Computing in Civil Engineering, 7(2):130–145 (1993).

11. Friswell, M.I. and J.E.T. Penny. "Is Damage Location Using Vibration Measurements Practical." http://www.swan.ac.uk/mecheng/staff/mfriswell/PDFfiles /DAMAS97.html, 1997.

12. Gehling, R.N. and others. Passive and Active Control of Space Structures. Technical Report, Martin Marietta Astronautics Group, 1991.

13. Gordon, R. The 12-Meter Truss Active Control Experiment Design, Analysis, and Open-Loop Testing. Technical Report, Flight Dynamics Directorate Wright Laboratory Air Force Systems Command Wright-Patterson AFB OH, 1992.

14. Inman, D.J. "Vibration with Control, Measurement, and Stability,". Department of Mechanical and Aerospace Engineering State University of New York at Buffalo.

15. Jain, A.K. *Fundamentals of Digital Image Processing*. Englewood Cliffs NJ: Prentice-Hall, 1989.

16. James, G.H. and others. "An Experimental Study of Frequency Response Function (FRF) Based Damage Assessment Tools," Proceedings of the 16th International Modal Analysis Conference, 151–157 (1998).

17. Jones, K. *Finite Element Model Updating Using Antiresonant Frequencies*. MS thesis, Air Force Institute of Technology, Wright-Patterson AFB OH, March 2000.

18. Juang, J. and R.S. Pappa. "An Eigensystem Realization Approach for Modal Parameter Identification and Model Reduction," Journal of Guidance and Control, 8(5):620–627 (1985).

19. Kudva, J. and others. "Damage Detection in Smart Structures Using Neural Networks and Finite Element Analysis," Proceedings of ADPA/AIAA/ASME/SPIE Conference on Active Materials and Adaptive Strutures, 559–562 (1991).

20. Leath, W.J. and D.C. Zimmermann. "Analysis of Neural Network Supervised Training with Application to Structural Damage Detection," Damage and Control of Large Structures, Proceedings of the 9th VPI& SU Symposium, 583–594 (1993).

21. Manning, R. "Damage Detection in Adaptive Structures Using Neural Networks," Proceedings of the of the 35th AIAA/ASME/ASCE/AHS/ASC Structures Structural Dynamics and Materials Conference, 160–172 (1994).

22. Meirovitch, L. *Elements of Vibration Analysis*. New York: McGraw Hill, 1986.

23. Oseguada, R.A. and Y. Qiang. "Damage Evaluation of Offshore Structures Using Resonant Frequency Shifts," Serviceability of Petroleum Process and Power Equipment ASME PVP, 239:31–37 (1992).

24. Rabiner, L.R. and B.H. Juang. "An Introduction to Hidden Markov Models," IEEE ASSP Magazine, 3(1):4–16 (January 1986).

25. Rabiner, L.R. and B.H. Juang. *Fundamentals of Speech Recognition*. Englewood Cliffs NJ: Prentice Hall, 1993.

26. Schulz, M.J. and others. "Frequency Response Function Assignment Technique for Structural Damage Identification," Proceedings of the 14th International Modal Analysis Conference, 105–111 (1996).

27. Stephens, J.E. and R.D. VanLuchene. "Integrated Assessment of Seismic Damage in Structures," Microcomputers in Civil Engineering, 9:119–128 (1994).

28. Sunstrand Data Control, Incorporated, 1500 N.E. 36th Street Redmond WA 98073. Sunstrand Data Control's Q-Flex Accelerometers.

29. Swenson, E. *Damage Detection Using Pattern Classifiers*. MS thesis, Air Force Institute of Technology, Wright-Patterson AFB OH, March 1998.

30. Vandiver, J.K. "Detection of Structural Failure on Fixed Platforms by Measurement of Dynamic Response," Journal of Petroleum Technology, 12:305–310 (March 1977).

31. Williams, E.J. and others. "A Frequency-Change Correlation Approach to Damage Detection," Proceedings of the 15th International Modal Analysis Conference, 652–657 (1997).

32. Worden, K. and others. "Neural Networks for Fault Location," Proceedings of the 11th International Modal Analysis Conference, 620–627

33. Wu, X. and others. "Use of Neural Networks in Detection of Structural Damage," Computers and Structures, 42(4):649–659 (1992).

34. Yuen, M.M.F. "A Numerical Study of the Eigenparameters of a Damaged Cantilever," Journal of Sound and Vibration, 103(3):301–310 (1985).

Vita

Dougas E. Gaeta was born on February 7, 1972 in Lynn, Massachusetts. He enlisted in the Massachusetts Army National Guard in 1990 immediately following graduation from high school. In addition to serving in the National Guard, he attended the University of Massachusetts in Amherst, MA where he graduated with a B.S. in Mechanical Engineering in May 1995, Cum Laude. He was selected for Officer Training School (OTS), and attended OTS at Maxwell AFB, AL. Officer Trainee Gaeta was commissioned in January 1996. Second Lieutenant Gaeta then went to Los Angeles AFB, CA where he served as a project officer in the Milstar Space and Ground Segment at the MILSATCOM Joint Program Office. In 1998, 1Lt Gaeta began his pursuit of a M.S. in Astronautical Engineering at the Air Force Institute of Technology. Upon completion of his studies, Captain Gaeta will be reassigned to the Engineering and Analysis Directorate, Space Warfare Center, Schriever AFB, CO.

Permanent address:	3 Democracy Drive
	Amesbury, MA 01913

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** 09 MAR 00	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE DAMAGE IDENTIFICATION IN A REAL STRUCTURE USING RESONANT AND ANTI-RESONANT FREQUENCIES	**5. FUNDING NUMBERS**
6. AUTHOR(S) Douglas E. Gaeta, Capt, USAF	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology 2950 P Street, BLDG 640 Wright-Patterson AFB, OH 45433-7765	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/GA/ENY/00M-02
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFIT/ENY 2950 P Street, BLDG 640 Wright-Patterson AFB, OH 45433-7765	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES
Advisor: Lt Col Jeffrey Turcotte
(937)255-3636, extension 4597
jeffrey.turcotte@afit.af.mil

12a. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

Several damage detection algorithms were developed and tested on a structure using 112 different damage conditions and one undamaged condition. The algorithms were trained using two frequency response functions (FRFs) from each damage case and then tested using the same two FRFs, but from newly acquired data. A data reduction technique from the field of speech recognition was adapted for this damage detection application. The data was reduced by greater than an order of magnitude, via a discrete, point-by-point, integration process in both training and testing. As shifts in resonant and anti-resonant frequencies were caused by the damage, another damage detection algorithm was developed that extracted the resonant and anti-resonant frequencies from the two FRFs. This algorithm vectorized the frequencies of the peaks and valleys in the FRFs for comparison. Over 99% accuracy was obtained using both the adapted speech recognition method and the resonant and anti-resonant frequencies method. However, only 44% accuracy was achieved by training the resonant and anti-resonant frequencies method with synthetic, Finite Element Model generated data, and testing it with measured data.

14. SUBJECT TERMS Damage Detection, Resonant Frequencies, Anti-Resonant Frequencies, Vibrations	**15. NUMBER OF PAGES** 93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL